Forord

Jeg vil aller først takke min veileder Olav Lysne for oppofrende, god, inspirerende og ikke minst hyggelig veiledning. Stor takk også til Bjørn Kirkerud for gjennomlesning og nyttige kommentarer den siste kritiske tiden og til Henrik for samvittighetsfull korrekturlesning den sommeren solen endelig viste seg i Oslo.

Jeg takker videre mine medstudenter, Tor, Henrik, Jo, Rune og mange andre ikke nevnte men heller ikke glemte, for diskusjonene om fotball og alt annet som gjør livet verdt å leve og som gjorde at jeg alltid gledet meg til å gå på Ifi. Takk også til Petya hvis humor og vennskap hjalp meg gjennom stundene jeg ikke kunne være tilstede på Ifi.

Ikke minst vil jeg til sist takke Cecilia, Miklós, Bence og Babkó uten hvis hjelp og kjærlige støtte denne oppgaven aldri ville blitt skrevet.

Oslo, 5. august 1994

Peter Csaba Ølveczky

“Gilgamesj, hvor løper du hen?

Det livet du søker, det finner du aldri.”

Fra *Gilgamesj* (sumerisk epos).

Om denne oppgaven

Oppgavens tema

Et datamaskinprogram går enten i evig tid eller stopper etter endelig tid. Det hadde vært ønskelig å kunne vite på forhånd hvorvidt et program stopper. Aller helst skulle vi hatt en algoritme (som selvfølgelig selv må stoppe!) som gitt et program klarer å avgjøre hvorvidt dette stopper for all input. Det er påvist at en slik algoritme ikke kan eksistere, og man er derfor fornøyd når en algoritme klarer å påvise terminering av så mange som mulig av de programmene som virkelig stopper. Programmer kan være skrevet i forskjellige språk og vi ønsker derfor å ha en måte å resonnere om programmer generelt. Ethvert program kan representeres av et *omskrivningssystem*, og det fins også programmeringsspråk der et program rett og slett *er* et omskrivningssystem. Å påvise at et omskrivningssystem terminerer er altså ikke noe annet enn å forsikre seg om at et program alltid stopper, og det er beskrevet mange teknikker som noen ganger kan brukes for å gjøre dette.

Typer spiller en stor rolle i programmeringsspråk og understøtter klarhet, letter feilfinning og lar oss arbeide med objekter av forskjellige slag i steden for med bitsekvenser. *Subtyper* og *arv* som begreper i programmeringsspråk ble introdusert i språket *Simula*, og er etterhvert blitt faste elementer i de fleste nyere språkene. *Typeordnet algebra* formaliserer disse begrepene, og *typeordnet omskrivning* generaliserer programmering med subtyper og multippel arv. Typeordnet algebra håndterer også ikke-veldefinerte uttrykk, som f. eks. divisjon med 0, som tidligere forårsaket problemer under forsøk på å gi algebraiske spesifikasjoner av abstrakte datatyper, på en elegant måte. Dette har også bidratt til den store interessen for typeordnet omskrivning, en interesse som også viser seg i at programmer i f. eks. språket *OBJ3* er typeordnede omskrivningssystemer.

I mangel på publiserte teknikker for å påvise terminering av typeordnede omskrivningssystemer, ser man bort fra typeinformasjon når man skal påvise at slike systemer stopper (se f. eks. implementasjonen av *ELIOS-OBJ* [Gn92b]). Dette svekker imidlertid styrken til algoritmene betraktelig, da det fins mange typeordnede omskrivningssystemer som terminerer, men der det samme omskrivningssystemet uten typer *ikke* gjør det. Anta f. eks. at Nat er en subtype av en type Int og la f og g være to funksjonssymboler der g har funksjonsprofilen $g : Nat \rightarrow Int$. Det typeordnede omskrivningssystemet bestående av regelen

$$f(x : Nat) \longrightarrow f(g(x))$$

vil terminere siden $g(x)$ er av typen Int. Ser man derimot bort i fra typene får man systemet $\{f(x) \longrightarrow f(g(x))\}$ som ikke terminerer. Det er imildertid ikke helt problemfritt å lage brukbare metoder som påviser terminering av typeordnede omskrivningssystemer ("typeordnet

terminering"). Anta nemlig at ovenstående system berikes med en subtype til Nat kalt $Zero$, at 0 er en konstant av type $Zero$ og at $g : Zero \rightarrow Zero$ er en ny funksjonsprofil. Det typeordnede omskrivningssystemet $\{f(x:Nat) \longrightarrow f(g(x))\}$ terminerer da ikke lenger siden det kan ha en uendelig omskrivningssekvens $f(0) \longrightarrow f(g(0)) \longrightarrow f(g(g(0)) \longrightarrow \cdots$. En annen vanskelighet som en algoritme som påviser terminering av det opprinnelige systemet støter på, er at er at dette systemet utvidet med en regel $g(x:Nat) \longrightarrow x$ ikke lenger er terminerende. De fleste metodene som viser terminering av utypede systemer (de som baserer seg på såkalte forenklingsordninger) ville nemlig ikke nøle med med å fastslå at et terminerende omskrivningssystem utvidet med en regel på formen $g(x) \longrightarrow x$ også er terminerende.

I denne oppgaven presenterer jeg en metode som er enkel å bruke for å påvise typeordnet terminering. Med denne metoden vil man kunne påvise at $\{f(x : Nat) \longrightarrow f(g(x))\}$ i det opprinnelige systemet nevnt over stopper, samtidig som den (selvfølgelig!) ikke viser at systemet utvidet med profilen $g : Zero \rightarrow Zero$ eller en regel $g(x:Nat) \longrightarrow x$ stopper.

Min fremgangsmåter består i å gjøre et typeordnet omskrivningssystem om til et system uten typer, og jeg kan deretter påvise terminering av det utypede systemet med kjente teknikker.

Oppgavens oppbygning

Denne oppgaven er delt opp i to deler. Hoveddelen, mitt arbeid innen typeordnet terminering, presenteres i *del II*. *Del I* kan ses på som en innledning, og starter med en kort innføring i utypet omskrivning (siden jeg i del II gjør om et typeordnet system til et utypet). Noen begreper herfra vil også brukes uforandret i typeordnet omskrivning. Del I fortsetter så med et kapittel som beskriver teknikker for å terminering av utypede omskrivningssystemer. Alle teknikkene som presenteres her vil kunne brukes i det typeordnede tilfellet. Min metode vil ha meget stor fordel av å kunne *kvasiordne* funksjonssymboler. Relativt stor del av kapitlet vies derfor kvasiordninger. De følgende kapitlene gir en kort innføring i hhv. typeordnet algebra, typeordnet omskrivning og typeordnet komplettering. I disse kapitlene motiverer jeg kravene om (pre-) regulære signaturer og typeminskende regler, to betingelser jeg vil sette til systemene i del II.

Man kan lure på hva typeordnet komplettering har med typeordnet terminering å gjøre. Strengt tatt kunne jeg utelatt kapitlet om typeordnet komplettering. Det er to grunner til at jeg likevel tar det med. For det første motiverer kapitlet i stor utstrekning hvorfor man hittil nesten bare har jobbet med typeminskende systemer. For det andre har jeg villet at del I også skulle tjene som en introduksjon til utypet terminering, typeordnet algebra og typeordnet omskrivning for eventuelle interesserte, og da hører typeordnet komplettering naturlig med. Denne ambisjonen har gjort del I lenger enn absolutt nødvendig, og jeg håper derfor at Anton Tsjekhovs ord "Kortheten er talentets mor" ikke kan brukes om denne oppgaven.

Del II av oppgaven beskriver først min metode, TT-metoden, for å påvise typeordnet terminering, etterfulgt av en presentasjon av tidligere arbeid innen området. Denne delen fortsetter så med en vurdering av min metode og forsøker å antyde hva man kan jobbe videre med for å forbedre denne. Oppgaven avsluttes med at jeg prøver å forutsi hvor man skal lete – og ikke minst: hvor man ikke skal lete – når man skal lage andre metoder for å påvise typeordnet terminering.

Innhold

I Utypet og typeordnet algebra og omskrivning **13**

1 Utypet algebra og omskrivning **15**

1.1 Utypet algebra 15

1.2 Utypet omskrivning 16

1.2.1 Egenskaper til omskrivningssystemer 17

1.2.2 Komplettering 17

1.2.3 Utvidelser av omskrivning 18

2 Terminering av utypet omskrivning **19**

2.1 Terminering 19

2.2 Termineringsordninger 20

2.3 Forenklingsordninger 23

2.3.1 Velfunderthet for avledninger 23

2.3.2 Omslutning 24

2.3.3 Forenklingsordninger 26

2.3.4 Stiordninger 27

2.3.5 Multiset stiordning 27

2.3.6 Leksikografisk stiordning 29

2.3.7 Stiordninger med variable 30

2.3.8 Ordningene som prosedyrer 30

2.3.9 Andre forenklingsordninger 31

2.4 Kvasiordninger og kvasiforenklingsordninger 32

2.4.1 Kvasiordninger 33

2.4.2 Kvasiforenklingsordninger 34

2.4.3 Leksikografisk stiordning 36

2.4.4 Multiset stiordning om igjen 37

2.4.5 Utvidelser av ordningene 39
2.4.6 Notasjonen videre 39

3 Typeordnet algebra **41**
3.1 Typeordnet signatur 43
3.2 Typeordnede algebraer 45
3.3 Initialitet og regularitet 47
3.3.1 Initialitet 47
3.3.2 Regularitet 48
3.3.3 Preregularitet 50
3.3.4 Monotoni 50
3.4 Likninger 51
3.4.1 Typene til termene i en likning 51
3.4.2 Gyldighet 52
3.4.3 Utledning av likninger 54
3.5 Oppsummering 55

4 Typeordnet omskrivning **57**
4.1 Den typeordnede omskrivningsrelasjonen 58
4.2 Typeminskende omskrivningssystemer (1) 60
4.3 Konfluens og konvergens 62
4.4 Retrakter 63
4.5 Oppsummering 64

5 Typeordnet komplettering **65**
5.1 Nødvendighet av typeminskning – Birkhoffs teorem og kritiske par 65
5.2 Kritiske par 67
5.2.1 Typeordnet unifikasjon 68
5.2.2 Kritiske par 68
5.3 Kompletteringsprosedyren 69
5.3.1 Korrekthet av inferenssystemet 70
5.4 Tomme typer 71
5.5 Typeminskning – enda en gang 72
5.6 Sammendrag 74

II Terminering av typeordnet omskrivning **75**

6 Speiling – en måte å påvise typeordnet terminering **77**

6.1 Terminering av typeordnede omskrivningssystemer 79

6.1.1 Definisjon av terminering – tomme typer 80

6.1.2 Retrakter og terminering 81

6.2 Speilinger 82

6.2.1 Syntaktisk speiling 82

6.3 Merking av funksjonssymbolene – beregning av $\lfloor\ \rfloor_F$ 85

6.3.1 Funksjonssymbolmengden F og funksjonen $\lfloor\ \rfloor_F$ – variant 1 86

6.3.2 En annen måte å definere F på 87

6.4 Funksjonssymbolsprang 88

6.5 Regelmengden R' i TT1-speilingen 91

6.6 Termineringsprosedyrer indusert av TT1-speilingen 94

6.7 Begrunnelser for speilingen 95

6.8 En svakhet med TT1 98

7 TT2-metoden – en forbedring av TT1-metoden **99**

7.1 TT2-*mpo* 99

7.2 Korrekthetsbevis for TT2-*mpo*-metoden 101

7.2.1 Speilingen 102

7.2.2 Terminering av $\mathcal{R}$ 104

7.3 Generalisering av TT2 105

8 TT3-metoden – en utvidelse av TT2-metoden **107**

8.1 Motivasjon og vanskeligheter 107

8.2 Nødvendigheten av TT3 109

8.3 Idéen bak TT3 110

8.4 Idéen dessverre bare bra for unære funksjonssymboler 112

8.4.1 Forandre stiordningene? 112

8.4.2 En ny ordning? 112

8.4.3 Bare unære funksjonssymboler kan bryte krav 7.1 113

8.5 TT3-metoden for stiordninger 113

8.5.1 Beregnbarhet av TT3-*mpo*-metoden 113

8.5.2 Eksempler på bruk av TT3-*mpo* 115

8.6 Korrekthetsbevis for TT3-*mpo* . . . 116
8.6.1 Speilingen $\langle F, \lfloor\ \rfloor, R''' \rangle$ . . . 116
8.6.2 Korrekthetsbeviset . . . 117
8.7 Terminering av omskrivningssystemer utvidet med retrakter . . . 120
8.8 Svakheter og mulige forbedringer av TT3-metoden . . . 121
8.8.1 Prosessen kan itereres – TT4, TT5, ... 121
8.8.2 En svakhet . . . 123
8.8.3 Mulige forbedringer . . . 124
8.8.4 TT3 for andre ordninger . . . 125

9 Utvidet Knuth-Bendix ordning 127
9.1 Knuth-Bendix-ordningen . . . 127
9.2 En modernisert Knuth-Bendix-ordning . . . 128
9.3 Utvidet *kbo* . . . 130
9.3.1 *ukbo*-likhet . . . 131
9.3.2 Korrekthetsbevis for *ukbo*-terminering . . . 132
9.3.3 Et eksempel . . . 134

10 TT-metodene basert på *ukbo* 135
10.1 TT1-*ukbo* . . . 135
10.2 TT2-*ukbo* . . . 136
10.3 TT3-*ukbo* . . . 137
10.3.1 Bevis av TT3-*ukbo*-terminering . . . 138
10.3.2 Beregnbarhet av TT3-*ukbo* . . . 141
10.3.3 Et eksempel . . . 142

11 Gnaedigs metode og regeltransformasjoner 143
11.1 Gnaedigs metode . . . 143
11.2 Regeltransformasjoner . . . 144

12 Vurdering og mulige forbedringer av TT-metoden 151
12.1 Fordeler . . . 151
12.2 Ulemper . . . 153
12.3 Utfordringer . . . 155

13 Nøkkelen til typeordnede termineringsordninger **157**

13.1 Håndtere variable 157

13.1.1 Regler for sammenligning av variable i termineringsordninger 158

13.1.2 Typeordnet *mpo* 158

13.1.3 Her er altså nøkkelen – og utfordringene 160

13.2 Her er det ingen nøkler 160

13.2.1 Ordninger over typer 160

13.2.2 Andre idéer som ikke må forfølges 162

14 Oppsummering **163**

Del I

Utypet og typeordnet algebra og omskrivning

Kapittel 1

Utypet algebra og omskrivning

For å vise terminering av et typeordnet omskrivningssystem vil jeg gjøre om dette til utypet omskrivningssystem. Dette kapitlet omhandler utypet omskrivning. Komplettering hører naturlig med til de fleste generelle fremstillinger av omskrivningsystemer. Jeg vil derfor gi en introduksjon til utypet komplettering i dette kapitlet, et kapittel som av hensyn til oppgavens omfang nødvendigvis må bli ganske kort.

1.1 Utypet algebra

En (utypet) *signatur* F er en tellbar mengde funksjonssymboler uten predefinert mening. Jeg vil[1] i denne oppgaven anta at hver $f \in F$ er tilordnet en *aritet*, et ikke-negativt heltall $ar(f)$. Et funksjonssymbol med aritet 0 kalles også en *konstant*. En F-*algebra* er et par $\mathcal{A} = (\mathcal{D}, F_{\mathcal{A}})$ hvor $\mathcal{D}$ er en mengde, kalt *domenet* til $\mathcal{A}$, og $F_{\mathcal{A}}$ er en mengde funksjoner slik at det for enhver $f \in F$ fins en funksjon $f_{\mathcal{A}} : \mathcal{D}^{ar(f)} \rightarrow \mathcal{D}$ i $F_{\mathcal{A}}$. Funksjonen $f_{\mathcal{A}}$ kalles *tolkningen* av f i $\mathcal{A}$. Jeg skriver ofte $\mathcal{A}$ også for domenet $\mathcal{D}$ til en algebra $\mathcal{A}$ for å unngå å introdusere unødvendig mange nye symboler.

En signatur F kan ha flere F-algebraer, og en funksjon $\phi : \mathcal{A} \rightarrow \mathcal{B}$ er en *homomorfi* fra en F-algebra $\mathcal{A}$ til en F-algebra $\mathcal{B}$ dersom

$$\phi(f_{\mathcal{A}}(a_1, \ldots, a_n)) = f_{\mathcal{B}}(\phi(a_1), \ldots, \phi(a_n))$$

for alle funksjonssymboler f i F og elementer $a_1, \ldots, a_n$ i $\mathcal{A}$.

For databehandlere er *termalgebraen* en viktig F-algebra, siden den i en viss forstand er den mest generelle F-algebraen. Gitt F og en mengde av variabelsymboler X disjunkt fra F defineres domenet $T(F, X)$ til termalgebraen $T(F, X)$ som den minste mengden $T(F, X)$ som oppfyller

$$\begin{array}{rcl} X & \subseteq & T(F, X) \\ t_1, \ldots, t_n \in T(F, X) \wedge f \in F & \Longrightarrow & f(t_1, \ldots, t_n) \in T(F, X) \text{ for } n = ar(f) \end{array}$$

Tolkningen $f_{T(F,X)}$ av et funksjonssymbol f i algebraen $T(F, X)$ er gitt ved

$$f_{T(F,X)}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$$

[1] pga. at den leksikografiske stiordningen vi skal se på senere forlanger dette

der høyresiden er et syntaktisk objekt, en *term*. Algebraen $T(F, \emptyset)$ av variabelfrie termer kalles *grunntermalgebraen* til F og er *initiell* i klassen av alle F-algebraer, det vil si at det fins én og bare én homomorfi ϕ fra $T(F, \emptyset)$ til enhver F-algebra $\mathcal{A}$. Mengden av variable i en term t skrives $\mathcal{V}(t)$.

En *algebraisk spesifikasjon* $\langle F, E \rangle$ består av en signatur F og en mengde *likninger* $E = \bigcup_{i=1}^{n} \{s_i \approx t_i\}$ der $s_i, t_i \in T(F, X)$ for en variabelmengde X. En likning $s \approx t$ er *gyldig* i en algebra $\mathcal{A}$ hvis og bare hvis $\phi(s) = \phi(t)$ for alle homomorfier $\phi : T(F, X) \rightarrow \mathcal{A}$, dvs. dersom s og t tolkes som samme element i $\mathcal{A}$ for alle tolkninger i $\mathcal{A}$ av variable i likningen. En F-algebra $\mathcal{A}$ er en *modell* for E når hver likning i E er gyldig i $\mathcal{A}$.

En interessant problemstilling er hvorvidt en kandidatlikning $s \approx t$ er gyldig i alle modeller til likningsmengde E, dvs. om $s \approx t$ er en *logisk konsekvens* av E. Jeg skriver $s \approx_E t$ når $s \approx t$ er en logisk konsekvens av E. Dette *gyldighetsproblemet* er ikke avgjørbart for alle spesifikasjoner (se f. eks. [Lin92]). For mange spesifikasjoner er det imidlertid avgjørbart, og man ønsker ofte å lage prosedyrer som løser gyldighetsproblemet for disse spesifikasjonene.

1.2 Utypet omskrivning

Anta gitt en termalgebra $T(F, X)$. En *omskrivningsregel* er et ordnet par av termer $l, r \in T(F, X)$ og skrives $l \longrightarrow r$. Et *omskrivningssystem* R er en (endelig eller uendelig) mengde av omskrivningsregler.

Omskrivningssystemer kan modellere ikke-deterministiske programmer og Turingmaskiner og dermed alle mulige programmer. Se bl. a. [Kir92], [Der82], [Der87] og [DJ90] for eksempler på bruk av omskrivningssystemer. Det fins også programmeringsspråk, f. eks. OBJ3 [GW88], der et program er et omskrivningssystem.

En term t kan sees på som et tre[2], og en *posisjon* p i t er en sekvens av heltall adskilt av tegnet . (punktum), der den tomme posisjonen skrives ϵ. Subtermen $t|_p$ defineres

$$\begin{aligned} t|_\epsilon &= t \\ f(t_1, \ldots, t_n)|_{i.p} &= t_i|_p \quad \text{der } i \leq n \end{aligned}$$

Termen $t[u]_p$ er termen t der subtermen $t|_p$ er byttet ut med termen u.

En *substitusjon* σ er en funksjon fra en variabelmengde Y inn i $T(F, X)$ og kan utvides på en éntydig måte til en homomorfi $\sigma^* : T(F, Y) \rightarrow T(F, X)$. Denne homomorfien σ^* skrives ofte σ. Siden den blir éntydig bestemt av verdien den tilordner variablene skrives den på formen $\sigma = \{y_1 \mapsto t_1, \ldots, y_n \mapsto t_n\}$. Jeg vil bruke postfiksnotasjon for substitusjoner. En substitusjon σ sies å være en *unifikator* for termene t og u dersom $t\sigma = u\sigma$.

Den binære omskrivningsrelasjonen $\xrightarrow[R]{}$ over $T(F, X)$ defineres ved at $t \xrightarrow[R]{} u$ hvis og bare hvis det fins en regel $l \longrightarrow r$ i R, en posisjon p i t og en substitusjon σ slik at $t|_p = l\sigma$ og $u = t[r\sigma]_p$. Jeg skriver av og til $t \xrightarrow[R]{l \rightarrow r} u$ for en omskrivning med en regel $l \longrightarrow r$. Den symmetriske relasjonen til $\xrightarrow[R]{}$ skrives $\xleftarrow[R]{}$ og den symmetriske tillukningen $\xleftrightarrow[R]{}$. For en vilkårlig binær relasjon $\simeq$ betegnes den transitive og refleksive-transitive tillukningen hhv. $\overset{+}{\simeq}$ og $\overset{*}{\simeq}$.

[2] som i databehandlingsverdenen alltid vokser nedover

1.2.1 Egenskaper til omskrivningssystemer

Én av de viktigste egenskapene et omskrivningssystem kan ha er *terminering*. R terminerer hvis og bare hvis det ikke fins noen uendelig omskrivningssekvens $t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots$ av termer i $T(F, X)$. Et eksempel på et ikke-terminerende system er $R = \{a \longrightarrow f(a)\}$ og et eksempel på et terminerende er $R' = \{f(a) \longrightarrow a\}$. Terminering av utypet omskrivning vil bli behandlet nærmere i kapittel 2.

En term t er *irredusibel* i R dersom det ikke fins noen term t' slik at $t \xrightarrow[R]{} t'$. Termen u er en *(R-)normalform* for en term t, som skrives $t \xrightarrow[R]{!} u$, dersom $t \xrightarrow[R]{*} u$ og u er irredusibel i R. Merk at en term kan ha 0, 1 eller flere normalformer. Et vilkårlig transitiv binær relasjon $\longrightarrow$ er *lokalt konfluent* dersom[3]

$$\longleftarrow \circ \longrightarrow \;\subseteq\; \xrightarrow{*} \circ \xleftarrow{*}$$

og et omskrivningssystem R er *konvergent* dersom R er både terminerende og lokalt konfluent. Enhver term t har nøyaktig én R-normalform når R er konvergent [KB70], og denne normalformen skrives $t!R$. Dessuten gjelder

$$\xleftrightarrow[R]{*} \;=\; \xrightarrow[R]{!} \circ \xleftarrow[R]{!}$$

for konvergente systemer R (se f. eks. [Kir92]), slik at man kan undersøke hvorvidt $t \xleftrightarrow[R]{*} u$ gjelder ved å sjekke om $t!R = u!R$.

1.2.2 Komplettering

Dette siste faktum kan benyttes til å lage en desisjonsprosedyre for gyldighetsproblemet i endel spesifikasjoner. Gitt en algebraisk spesifikasjon $\langle F, E\rangle$, og la regelmengden R_E være gitt $\{v \longrightarrow h \mid v \approx h \in E\}$. Birkhoffs teorem (sitert i [Kir92]) sier da

$$\approx_E \;=\; \xleftrightarrow[R_E]{*}$$

som medfører at når man klarer å lage et konvergent omskrivningssystem R slik at $\xleftrightarrow[R]{*} = \xleftrightarrow[R_E]{*}$, så gjelder $\approx_E \;=\; \xrightarrow[R]{!} \circ \xleftarrow[R]{!}$. Da er $t \approx_E u$ hvis og bare hvis $t!R = u!R$ og vi har en desisjonsprosedyre for gyldighetsproblemet.

En *kompletteringsprosedyre* er en prosedyre som gitt en algebraisk spesifikasjon $\langle F, E\rangle$ forsøker å konstruere et konvergent omskrivningssystem R slik at $\approx_E \;=\; \xleftrightarrow[R]{*}$. Siden gyldighetsproblemet generelt ikke er avgjørbart klarer ikke en kompletteringsprosedyre alltid å lage en slik R.

En mye brukt kompletteringsprosedyre er den såkalte *Knuth-Bendix-prosessen* (KB-prosessen) først beskrevet av Donald Knuth og Peter Bendix i [KB70]. Denne prosessen starter (hvis mulig) med å lage et terminerende system R slik at $\approx_E \;=\; \xleftrightarrow[R]{*}$. R er imidlertid ikke nødvendigvis konfluent. Det viser seg at dersom R ikke er konfluent, så er det fordi en instans $l\sigma$ av en venstreside til en regel $l \longrightarrow r$ i R kan skrives om til forskjellige termer t og u (der (t, u) kalles et *kritisk par*) som har ulike normalformer. Det kritiske paret blir forsøkt

[3] Jeg vil benytte meg av at en n-ær relasjon $\mathcal{R}$ kan ses på som en mengde $\mathcal{R}$ av n-tupler der $\mathcal{R}(a_1, \ldots, a_n) \iff \langle a_1, \ldots, a_n\rangle \in \mathcal{R}$, og skriver $\mathcal{R}_1 \circ \mathcal{R}_2$ for sammensetningen av to relasjoner, dvs. $a\ \mathcal{R}_1 \circ \mathcal{R}_2\ b \iff \exists c \mid a\mathcal{R}_1 c \wedge c\mathcal{R}_2 b$.

lagt til R slik at terminering og $\approx_E = \overset{*}{\underset{R}{\longleftrightarrow}}$ hele tiden er invariant i R. På denne måten fortsetter prosessen med å finne kritiske par og lage nye regler helt til enten R er konvergent eller R må utvides med en regel som gjør at terminering ikke lenger kan garanteres. Det tredje mulige utfallet er at prosessen går i det uendelige og hele tiden genererer nye kritiske par og regler. Når det første tilfellet slår til, har man den ønskede desisjonsprosedyren og i det andre tilfellet har man bare fått en mengde gyldige regler. I det tredje tilfellet utgjør prosessen en semidesisjonsprosedyre for gyldighetsproblemet på den måten at alle gyldige likninger før eller siden lar seg bevise. Se f. eks. [KB70], [Kir92], [Lin92] og [DJ90] for mer detaljerte beskrivelser av KB-prosessen.

1.2.3 Utvidelser av omskrivning

Av og til ønsker man at en omskrivning med en regel $l \longrightarrow r$ bare skal foretas dersom de aktuelle verdiene til variablene i l og r oppfyller visse kriterier. Eksempelvis kunne

$$\textbf{if } \mathit{empty}(x) \approx_E \mathit{false} \textbf{ then } \mathit{push}(\mathit{top}(x), \mathit{pop}(x)) \longrightarrow x$$

vært en regel under operasjon på stakker. Omskrivning med slike regler kalles *betinget omskrivning*. Betinget omskrivning er omfattet av stor interesse men byr også på mange problemer. Vi har f. eks. sett at det generelt ikke avgjørbart hvorvidt betingelsen slår til da den ofte er på formen $s \approx_E t$.

Komplettering kan mislykkes fordi man må legge til en regel $l \longrightarrow r$ som gjør at terminering ikke lenger kan garanteres. Et eksempel på en slik regel er $f(x, y) \longrightarrow f(y, x)$. I *utvidet komplettering* legger man ikke slike regler til R, men har dem i stedet "i bakhodet" hele tiden (i form av en likningsmengde E'). Dette har ført til et nytt forskningsområde innen omskrivning, *omskrivning modulo en kongruens*.

Jeg vil ikke se på dise utvidelsene av omskrivning senere i oppgaven.

Kapittel 2

Terminering av utypet omskrivning

Et omskrivningssystem *terminerer* dersom det ikke kan forekomme uendelige omskrivningssekvenser i systemet. I og med at terminering er én av de viktigste egenskapene ved et omskrivningssystem, ønsker vi å kunne avgjøre algoritmisk om et gitt system terminerer. Siden omskrivningssystemer kan representere Turing-maskiner dreier termineringsproblemet seg egentlig om hvorvidt programmer stopper. Da dette stoppeproblemet ikke er avgjørbart kan det heller ikke finnes noen desisjonsprosedyre som kan fastslå hvorvidt et gitt omskrivningssystem terminerer eller ikke. Det beste man kan håpe på er ha en prosedyre som kan fastslå terminering av flest mulig av systemene som terminerer.

Dette kapitlet viser teknikker for å lage prosedyrer som kan påvise terminering av utypede systemer. Min fremgangsmåte for å vise typeordnet terminering, vil bygge på disse teknikkene. Jeg eksemplifiserer teknikkene med de to kanskje vanligste "termineringsordningene", *multiset* og *leksikografisk stiordning*. Det er også disse to jeg vil bruke i senere kapitler for illustrere og begrunne mine generelle metoder for å påvise terminering av typeordnede systemer.

Innholdet i dette kapitlet bygger i stor grad på Dershowitz' [Der87].

2.1 Terminering

Definisjon 2.1 (Terminering) *Et omskrivningssystem R* terminerer *for en mengde termer $T(F, X)$ dersom det ikke finnes en uendelig omskrivningssekvens*

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} t_3 \xrightarrow[R]{} \cdots$$

av termer i $T(F, X)$. Et omskrivningssystem er ikke-terminerende *dersom det finnes en slik uendelig omskrivningssekvens.*

Som nevnt hadde det vært ønskelig å kunne avgjøre på én eller annen måte hvorvidt et system terminerer eller ikke, men det viser seg imidlertid at Turingmaskiner kan simuleres av omskrivningssystemer. Siden det er umulig å avgjøre hvorvidt en Turingmaskin stopper fører det naturlig nok til at det umulig å avgjøre hvorvidt et omskrivningssystem terminerer.

Teorem 2.2 ([Der87]) *Det er ikke avgjørbart hvorvidt et omskrivningssystem terminerer, selv om det bare har to regler.*

Det har nylig blitt vist (sitert i [Pla93]) at det også er uavgjørbart hvorvidt et system med bare én regel terminerer. Dette betyr at det ikke kan finnes noen algoritme som gitt et system avgjør om systemet er terminerende eller ikke-terminerende. Det beste man kan håpe på er å finne algoritmer som påviser terminering for en del terminerende systemer, men ikke alle. Det er slike algoritmer jeg skal se på nedenfor.

Uendelige systemer?

Det har tidligere vært vanlig med et upresist forhold til størrelsen på signaturer og regelmengder. Eksempelvis skriver Dershowitz i [Der82] og kapittel 1 i [Der87] at et omskrivningssystem er en *endelig* mengde omskrivningsregler hvorpå det i kapittel 2 i den siste av disse artiklene står at et omskrivningssystem består av endelig eller *uendelig* mange regler.

Det har i de siste årene vært forsket endel rundt uendelige systemer, og jeg ville også veldig gjerne ha brukt et uendelig system i et bevis i kapittel 8. Jeg vil derfor anta at regelmengdene kan være uendelige og vil heller ikke legge andre begrensninger enn tellbarhet på størrelsen av funksjonssymbolmengden F.

Disse uklarhetene har ført til en viss begrepsforvirring i litteraturen. Eksempelvis omtales leksikografisk stiordning ofte som velfundert, selv om den bare er *velfundert for avledninger*, et uttrykk som defineres i avsnitt 2.3.

De fleste av metodene som presenteres nedenfor kan også benyttes for uendelige regelmengder og signaturer, mens alle metodene vil være spesielt egnet dersom enten regelmengden eller signaturen er endelig.

2.2 Termineringsordninger

Den intuitive måten å påvise terminering er å bruke såkalte "progresjonsfunksjoner" τ som er slik at dersom $t \xrightarrow[R]{} t'$ så er $\tau(t) > \tau(t')$ samtidig som τ-verdiene ikke kan gå "nedover" i det uendelige. Det er nødvendig å definere noen hjelpebegreper for å uttrykke dette.

Definisjon 2.3 ((Strikt) partiell ordning) *En binær relasjon $\succ$ over en mengde S er en (strikt[1]) partiell ordning over S dersom $\succ$ er ikke-refleksiv (dvs. $\neg\exists s \in S | s \succ s$) og transitiv (dvs. $\forall x, y, z \in S | x \succ y \wedge y \succ z \Longrightarrow x \succ z$).*

En partielt ordnet mengde $(S, \succ)$ består av en mengde S og en strikt partiell ordning $\succ$ over S.

Definisjon 2.4 (Velfunderthet) *En (strikt) partielt ordnet mengde $(S, \succ)$ er* velfundert *dersom det ikke finnes noen uendelig $\succ$-minskende sekvens*

$$s_1 \succ s_2 \succ s_3 \succ \cdots$$

[1] Noen steder er begrepet partiell ordning definert ved refleksivitet, antisymmetri og transitivitet. Dette vil jeg senere kalle en *refleksiv partiell ordning*.

av elementer i S.

Eksempel 2.1 De naturlige tallene $\mathcal{N}$ danner en velfundert partielt ordnet mengde under ordningen $>$ fordi det ikke fins noen $>$-minskende sekvens som går forbi 0.

Idéen som ofte brukes for å vise terminering går ut på at $\tau : T(F,X) \rightarrow S$ er en *progresjonsfunksjon* som til enhver term tilordner en verdi i en velfundert partielt ordnet mengde $(S, \succ)$ slik at dersom $t \xrightarrow[R]{} t'$ så er $\tau(t) \succ \tau(t')$. Det kan da ikke eksistere en uendelig omskrivningssekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} t_3 \xrightarrow[R]{} \cdots$$

fordi det da ville eksistert en uendelig S-sekvens

$$\tau(t_1) \succ \tau(t_2) \succ \tau(t_3) \succ \cdots$$

som er umulig da $(S, \succ)$ er antatt å være velfundert.

Eksempel 2.2 Anta at omskrivningssystemet R består av regelen

$$f(x) \longrightarrow g(x)$$

La progresjonsfunksjonen $\tau : T(F,X) \rightarrow \mathcal{N}$ defineres ved at $\tau(t)$ er lik antall forekomster av symbolet f i t. Dersom $t \xrightarrow[R]{} t'$ så gjelder $\tau(t) > \tau(t')$. Siden $(\mathcal{N}, >)$ er velfundert har vi påvist terminering av R.

En funksjon $\tau : T(F,X) \rightarrow S$ og en partiell ordning $>$ over S kan gjøres om til en partielt ordnet mengde $(T(F,X), \succ)$ ved at $t \succ t'$ hvis og bare hvis $\tau(t) > \tau(t')$.

Følgende lemma ligger implisitt i bl. a. [Der87] og gjør at det (foreløpig) ikke er nødvendig å se på hele termalgebraen $T(F,X)$, men at det tilstrekkelig å påvise terminering av et omskrivningssystem for termer i grunntermalgebraen $T(F,\emptyset)$.

Lemma 2.5 *Anta at en signatur F inneholder minst én konstant, og la R være en regelmengde over termer i $T(F,X)$. Dersom det fins en uendelig omskrivningssekvens*

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} t_3 \xrightarrow[R]{} \cdots$$

av termer i $T(F,X)$, så fins det også en uendelig omskrivningssekvens

$$t'_1 \xrightarrow[R]{} t'_2 \xrightarrow[R]{} t'_3 \xrightarrow[R]{} \cdots$$

av grunntermer i $T(F,\emptyset)$.

Bevis: Det er bare å erstatte enhver variabelforekomst i den første omskrivningssekvensen med en konstant, så får man en tilsvarende uendelig omskrivningssekvens av grunntermer. □

Dermed er det tilstrekkelig å vise terminering av systemer der variablene i reglene rangerer over alle grunntermer. Jeg vil derfor skrive $T(F)$ for grunntermalgebraen $T(F,\emptyset)$ for å understreke at forskjellen mellom $T(F,X)$ og $T(F,\emptyset)$ ikke er av stor betydning når det gjelder terminering. Merk at reglene fremdeles består av termer i $T(F,X)$.

Teorem 2.6 ([Der87]) *Et omskrivningssystem R terminerer for en mengde termer $T(F)$ hvis og bare hvis det finnes en velfundert ordning $\succ$ over $T(F)$ slik at*

$$t \xrightarrow[R]{} t' \Longrightarrow t \succ t'$$

for alle termer t, t' i $T(F)$.

Eksempel (forts. av eks. 2.2) Definér ordningen $\succ$ slik at $t \succ t'$ hvis og bare hvis antall f'er i t er større enn antall f'er i t'. Siden $\xrightarrow[R]{} \subseteq \succ$ og $\succ$ er velfundert så terminerer R.

Problemet er at det (oftest) fins uendelige mange omskrivninger $t \xrightarrow[R]{} t'$, og for å unngå å måtte foreta uendelige mange tester for $\xrightarrow[R]{} \subseteq \succ$ trenger man følgende egenskaper for termineringsordninger:

Definisjon 2.7 (Monotoni) *En (strikt) partiell ordning $\succ$ over en mengde $T(F)$ er* monoton *dersom*

$$t \succ t' \Longrightarrow f(\ldots, t, \ldots) \succ f(\ldots, t', \ldots)$$

for alle termer t, t' i $T(F)$ og alle funksjonssymboler f i F.

Når en partiell ordning $\succ$ er monoton, betyr det at dersom $l\sigma \succ r\sigma$ så er også $t[l\sigma]_p \succ t[r\sigma]_p$ fordi monotoniegenskapen kan itereres. Klarer man å vise $l\sigma \succ r\sigma$ for alle instanser $l\sigma \longrightarrow r\sigma$ av hver regel $l \longrightarrow r$ gjelder derfor $t \succ t'$ for alle omskrivninger $t \xrightarrow[R]{} t'$. Dette kan brukes til å vise terminering på ved følgende klassiske teorem:

Teorem 2.8 ([Der87]) *Et omskrivningssystem R terminerer for en termmengde $T(F)$ hvis og bare hvis det fins en monoton velfundert partiell ordning $\succ$ over $T(F)$ slik at*

$$l\sigma \succ r\sigma$$

for alle grunnsubstitusjoner σ av variable i l til enhver regel $l \longrightarrow r$ i R.

Dette er den enerådende fremgangsmåten for å påvise terminering. Problemet ligger i å finne en passende velfundert monoton partiell ordning $\succ$, og det er dette den videre fremstillingen skal handle om. Det er opplagt at en slik ordning $\succ$ må være transitiv og ikke-refleksiv. Grunnen til at man ikke forlanger at den er *total* er at det kan være hendig å definere den så lite restriktivt som mulig til å begynne med, og at ordningen kan "bygges ut" underveis etter behov. En total ordning kan heller ikke håndtere alle regler med variable, da hverken $f(x, y) \succ f(y, x)$ eller $f(y, x) \succ f(x, y)$ bør gjelde.

Jeg vil bruke begrepet *termineringsordning* for en ordning $\succ$ som brukes til å vise terminering, og jeg vil skrive at et ordnet par av termer $\langle t, t' \rangle$ er $\succ$-minskende dersom $t \succ t'$.

2.3 Forenklingsordninger

Det kan ofte være vanskelig å finne en velegnet partiell ordning $\succ$ over termer og minst like vanskelig å vise at den er velfundert (se f. eks. originalbeviset for velfunderthet av Knuth-Bendix-ordningen i [KB70]). Tidligere måtte man for enhver ordning vise at den var velfundert.

Dershowitz gikk "den motsatte veien" og forsøkte se nærmere på hva som kjennetegner et ikke-terminerende system. Ved å finne et slikt kjennetegn, nemlig *selvomslutning*, definerte han en klasse ordninger, de såkalte *forenklingsordningene*, som alltid "oppdager" selvomslutning og dermed også (mulig) ikke-terminering. Forenklingsordninger gir mye enklere test på velfunderthet og har pga. dette ført til en veritabel oppblomstring av nye termineringsordninger, som f. eks. "path of subterms ordering" [Rus87], "recursive decomposition ordering" [JLR83] og semantisk stiordning [Der87].

2.3.1 Velfunderthet for avledninger

Forenklingsordninger bygger (bl. a.) på prinsippet at ordningen $\succ$ ikke nødvendigvis trenger å være velfundert for hele $T(F)$ (eller S dersom man bruker "eksplisitte" progresjonsfunksjoner over et domene S), men bare for omskrivningssekvensene som virkelig kan forekomme. Følgende eksempel illustrerer dette:

Eksempel 2.3 La R også denne gangen[2] bestå av regelen $f(x) \longrightarrow g(x)$. Vi definerer progresjonsfunksjonen $\tau : T(F) \rightarrow \mathcal{Z}$ der $\mathcal{Z}$ er mengden av heltall ved

$$\tau(t) = \text{antall forekomster av } f \text{ i } t \textit{ minus} \text{ antall forekomster av } g \text{ i } t$$

Den tilhørende ordningen er vanlig $>$, "større enn". Ordningen $\succ$ kan da på vanlig måte defineres $t \succ t' \Longleftrightarrow \tau(t) > \tau(t')$. Relasjonen $>$ er imidlertid ikke velfundert over $\mathcal{Z}$ fordi det fins uendelige $>$-minskende sekvenser i $\mathcal{Z}$ som f. eks. $2 > 1 > 0 > -1 > -2 > -3 > \cdots$ (tilsvarende fins uendelige $\succ$-sekvenser $f(f(a)) \succ f(f(g(a))) \succ f(g(a)) \succ f(g(g(a))) \succ f(g(g(g(a)))) \succ \cdots$).

Poenget er imidlertid at for enhver avledningssekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} t_3 \xrightarrow[R]{} \cdots$$

er den minste verdien til en $\tau(t_i)$ begrenset, fordi $\tau(t_i)$ alltid er større enn $-$(størrelsen av t_1) siden størrelsen på termene ikke blir forandret av et R-omskrivningssteg. For én omskrivningssekvens kan minste τ-verdi kanskje være -1000, for en annen -2 (f. eks. $f(f(a)) \xrightarrow[R]{} f(g(a)) \xrightarrow[R]{} g(g(a))$), for en tredje kanskje -10, men ingen omskrivningssekvens kan ha minste τ-verdi som ikke er begrenset nedad. Fordi $\xrightarrow[R]{} \subseteq \succ$ kan det da ikke finnes en uendelig omskrivningssekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} t_3 \xrightarrow[R]{} \cdots$$

Dersom relasjonen $\succ \cap \xrightarrow[R]{+}$ er velfundert for endelige regelmengder R, sier vi at $\succ$ er *velfundert for avledninger.*

[2] Vi har tidligere sett en mye enklere måte å vise terminering av R på.

2.3.2 Omslutning

For å komme frem til egenskaper ved terminerende systemer, kan man finne egenskaper som gjelder for alle ikke-terminerende systemer. Et første tips på en slik egenskap er at enhver uendelig omskrivningssekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots \xrightarrow[R]{} t_j \xrightarrow[R]{} \cdots \xrightarrow[R]{} t_k \xrightarrow[R]{} \cdots$$

er *gjentagende*, dvs. at en t_j er en subterm av en t_k for $j < k$. Det er opplagt at et gjentagende system ikke er terminerende, men det fins faktisk ikke-terminerende omskrivningssystemer som *ikke* har gjentagende omskrivningssekvenser. Et eksempel på et slikt system fins i [Der87].

Det er med andre ord ikke tilstrekkelig å vise at et omskrivningssystem ikke er gjentagende for å fastslå at det terminerer. Vi må derfor finne et annet kriterium for ikke-terminering.

Definisjon 2.9 (Omslutning) *En term* $t = f(t_1, t_2, \ldots, t_m)$ omslutter *en term* $t' = g(t'_1, t'_2, \ldots, t'_n)$, *som skrives*

$$t \trianglerighteq t'$$

dersom enten

$$\exists i \in 1..m \mid t_i \trianglerighteq t'$$

eller

$$f = g \wedge \forall i \in 1..n \mid t_i \trianglerighteq t'_i$$

Faktumet $t \trianglerighteq t'$ (som også kan skrives $t' \trianglelefteq t$) sier at t' er inneholdt i t på den måten at t' kan fås fra t ved å "fjerne" noen operatorer. Eksempelvis gjelder[3] $f(g(f(x))) \trianglerighteq f(f(x))$. Vi skriver $t \triangleright t'$ når $t \trianglerighteq t'$ og $t \neq t'$.

I en uendelig sekvens av termer bygget opp av endelige mange ulike funksjons- og variabelsymboler er det ikke overraskende at noen "mønstre" nødvendigvis må gjentas i sekvensen:

Teorem 2.10 (Kruskal, sitert i [Der87]) *Dersom* F *er en endelig mengde funksjonssymboler inneholder enhver uendelig sekvens*

$$t_1, t_2, \ldots, t_j, \ldots, t_k, \ldots$$

av termer i $T(F)$ *to termer* t_j *og* t_k *(der* $j < k$*) slik at* $t_j \trianglelefteq t_k$.

Definisjon 2.11 (Selvomsluttende avledninger) *En avledning*

$$t_1 \longrightarrow t_2 \longrightarrow \cdots \longrightarrow t_j \longrightarrow \cdots \longrightarrow t_k \longrightarrow \cdots$$

er selvomsluttende *dersom* $t_j \trianglelefteq t_k$ *for et par* $\langle j, k \rangle$ *der* $j < k$. *Et omskrivningssystem er selvomsluttende dersom den kan ha en selvomsluttende avledning.*

Enhver uendelig omskrivningssekvens må da være selvomsluttende dersom den inneholder et endelig antall ulike funksjonssymboler.

[3] Siden omslutning bare dreier seg om formen på termene, kan en variabel i denne sammenheng ses på som en konstant.

Teorem 2.12 ([Der87]) *Dersom R er et endelig omskrivningssystem så er R selvomsluttende hvis R er ikke-terminerende over $T(F)$.*

Bevis: Anta at R ikke terminerer. Da fins det en uendelig sekvens av termer

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots \xrightarrow[R]{} t_j \xrightarrow[R]{} \cdots \xrightarrow[R]{} t_k \xrightarrow[R]{} \cdots$$

Når R er endelig er termene $t_1, t_2, \ldots$ også bygget opp av et endelig antall funksjonssymboler, nemlig de som forekommer i t_1 og R. Teorem 2.10 gir da $t_j \trianglelefteq t_k$ for noen $j < k$. □

Vi kan også tillate uendelige systemer hvis mengden F av funksjonssymboler er endelig.

Korollar 2.13 *Dersom R er et omskrivningssystem over $T(F)$ der F er en endelig mengde, så fører ikke-terminering av R til at R er selvomsluttende.*

Bevis: Korollaret er en opplagt konsekvens av teorem 2.10. □

Selvomslutning kjennetegner altså et ikke-terminerende system, og er et av de "beste" kjennetegnene vi har. Det er et litt for sterkt i det henseende at det fins systemer som *både* er selvomsluttende og terminerende.

Eksempel 2.4 ([Der87]) $R = \{f(f(x)) \longrightarrow f(g(f(x)))\}$ er selvomsluttende og terminerende.

Det faktum at et endelig system R er selvomsluttende dersom det er ikke-terminerende kan nå brukes til å finne en partiell ordning $\succ$ som oppdager terminering. Anta at vi klarer å lage ordningen $\succ$ slik at $\rhd \subseteq \succ$ og $\xrightarrow[R]{} \subseteq \succ$. Klarer vi dette, har vi faktisk påvist terminering, fordi hvis R *ikke* terminerte, måtte det ha eksistert en uendelig sekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots \xrightarrow[R]{} t_j \xrightarrow[R]{} \cdots \xrightarrow[R]{} t_k \xrightarrow[R]{} \cdots$$

av grunntermer. Når R er en endelig regelmengde er denne sekvensen selvomsluttende, slik at $t_j \trianglelefteq t_k$. Siden $\xrightarrow[R]{} \subseteq \succ$ og $\succ$ er en transitiv ordning, har vi $t_j \succ t_k$. Men $t_j \trianglelefteq t_k$ fører til enten $t_j = t_k$, som er umulig da $\succ$ er ikke-refleksiv, eller til $t_j \lhd t_k$, som leder til selvmotsigelse da $\rhd \subseteq \succ$ gir $t_k \succ t_j$, samtidig som vi tidligere hadde $t_j \succ t_k$. Transitivitet og ikke-refleksivitet av $\succ$ utelukker denne muligheten. Betingelsene $\xrightarrow[R]{} \subseteq \succ$ og $\rhd \subseteq \succ$ sikrer altså at selvomsluttende avledninger ikke kan forekomme.

For å definere en slik $\succ$ hadde det mest ønskelige kanskje vært å bruke relasjonen $\rhd$ direkte, men det er dessverre uavgjørbart hvorvidt et system er selvomsluttende [Der87]. Vi må altså bruke en annen metode som gir oss $\rhd \subseteq \succ$. Hvis $\succ$ er en *forenklingsordning* oppnår vi dette.

2.3.3 Forenklingsordninger

For å kunne bruke metoden fra teorem 2.8 må $\succ$ være monoton. En forenklingsordning er en monoton utvidelse av relasjonen $\rhd$.

Definisjon 2.14 (Forenklingsordning) *En monoton partiell ordning* $\succ$ *er en* forenklingsordning *over en mengde termer* $T(F)$ *dersom den innehar* subtermegenskapen

$$f(\ldots, t, \ldots) \succ t$$

for alle termer i $T(F)$.

Lemma 2.15 ([Der82]) *For enhver forenklingsordning* $\succ$ *gjelder* $\rhd \subseteq \succ$.

Siden en forenklingsordning $\succ$ sikrer mot selvomslutning fører altså $\underset{R}{\longrightarrow} \subseteq \succ$ til at det ikke kan finnes en selvomsluttende omskrivningssekvens og at et endelig system R da må være terminerende.

Teorem 2.16 ([Der82]) *Et* endelig *termomskrivningssystem* R *terminerer for en mengde* $T(F)$ *dersom det fins en forenklingsordning* $\succ$ *over* $T(F)$ *slik at*

$$l\sigma \succ r\sigma$$

for alle regler $l \longrightarrow r$ *i* R *og for alle substitusjoner* σ *av variablene i* l.

Teoremet er også gyldig selv om R ikke er endelig sålenge F er det. Det kan synes merkelig å ville påvise terminering av uendelige systemer, men i teoretiske betraktninger og under komplettering kunne det f. eks. vært nyttig å definere R som mengden av alle mulige regler $t \longrightarrow t'$ som oppfyller $t \succ t'$.

En svakhet med fremgangsmåten indikert over er at det generelt fins uendelig mange substitusjoner σ som man må vise $l\sigma \succ r\sigma$ for. Derfor vil jeg også litt senere vise hvordan ordningene kan behandle termer med variable slik at $l \succ r$ medfører $l\sigma \succ r\sigma$ for alle substitusjoner σ.

Dershowitz deler inn termineringsordningene i to klasser. De *syntaktiske ordningene* blir indusert av en ordning $\succ_F$ (eller kvasiordning $\succsim_F$) over funksjonssymboler mens de *semantiske ordningene* bygger på en ordning $\succ_T$ (eller $\succsim_T$) over termer. Knuth-Bendix-ordningen, *kbo*, ble definert i [KB70] og bygger på både en ordning $\succ_T$ over termer[4] og en presedens $\succ_F$ over funksjonssymboler. Ordningen *kbo* er, som jeg senere skal vise, interessant i typeordnet omskrivning, der *kbo* kan brukes på en klasse regler stiordningene ikke klarer å håndtere. Jeg vil derfor definere og "modernisere" *kbo* i et eget kapittel (kapittel 9).

[4] ved $t \succ_T t'$ hvis og bare hvis $vekt(t) > vekt(t')$.

2.3.4 Stiordninger

De mest brukte syntaktiske ordningene er såkalte stiordninger. En stiordning $\succ_{po}$ bygger på en presedens $\succ_F$ over funksjonssymboler og er definert rekursivt ved

$$\frac{t_i \succeq_{po} t'}{f(\ldots, t_i, \ldots) \succ_{po} t'}$$

der $\succeq = (\succ \cup =)$ som sikrer subtermkravet og

$$\frac{f \succ_F g \qquad \forall i \in 1..n | f(t_1, \ldots, t_m) \succ_{po} t'_i}{f(t_1, \ldots, t_m) \succ_{po} g(t'_1, \ldots, t'_n)}$$

og

$$\frac{\langle t_1, \ldots, t_n \rangle \succ'_{po} \langle t'_1, \ldots, t'_n \rangle \qquad \forall i \in 1..n | f(t_1, \ldots, t_n) \succ_{po} t'_i}{f(t_1, \ldots, t_n) \succ_{po} f(t'_1, \ldots, t'_n)}$$

der $\succ'_{po}$ er en ordning på term-tupler indusert av $\succ_{po}$ som er slik at $\succ'_{po}$ er velfundert dersom $\succ_{po}$ er det. Den andre premissen til tredje regel sikrer at ingen "utestet" subterm t'_i er større enn $f(t_1, \ldots, t_m)$. Man kan spesielt merke seg at $t \succ_{po} t'$ gjelder dersom t inneholder et funksjonssymbol som er større i $\succ_F$ enn alle funksjonssymbolene i t'. Sammenligningen $\succ'_{po}$ kan være definert på forskjellige måter. De to vanligste er multiset sammenligning og leksikografisk sammenligning.

2.3.5 Multiset stiordning

Jeg skal i dette avsnittet definere én av de første ordningene som bygget direkte på begrepet "forenklingsordning", nemlig den partielle ordningen *multiset stiordning*, som (i modernisert utgave) fremdeles er én av de mest brukte termineringsordningene.

Et multiset[5] kan sees på som en "mengde" der samme element kan forekomme flere ganger, men der rekkefølgen av elementene ikke har noe å si. Formelt sett er et multiset M over en mengde S en funksjon

$$M : S \rightarrow \mathcal{N}$$

der $M(s)$ sier hvor mange ganger elementet s forekommer i multisettet M. Denne gir at to multiset er like hvis alle elementer i S forekommer like mange ganger i begge multisettene. Eksempelvis er multisettet $\{a, b, b\}$ lik multisettet $\{b, a, b\}$ men forskjellig fra $\{a, b\}$. Jeg bruker symbolet $\emptyset$ for det tomme multisettet (dvs. $\emptyset = \lambda s \in S : 0$) og symbolet $\setminus$ for multisetdifferanse (dvs. $M \setminus N = \lambda s \in S : \max(0, M(s) - N(s))$).

En partiell ordning over enkeltelementer induserer en partiell ordning over multiset på følgende måte:

Definisjon 2.17 (Multisetutvidelse av partiell ordning) *Gitt en mengde S og en partiell ordning $\succ$ over S defineres multisetutvidelsen $\succ^{multi}$ av $\succ$ over S-multiset rekursivt ved*

[5] Det engelske ordet *multiset* synes å være innarbeidet i norsk, slik at jeg velger å bruke det uoversatt videre i oppgaven.

multi-1: $$\{s_1, \ldots, s_n\} \succ^{multi} \emptyset \qquad \text{for } n \geq 1$$

multi-2: $$\frac{\{s_1, \ldots, s_m\} \setminus \{s_i\} \succeq^{multi} \{t_1, \ldots, t_n\} \setminus \{t_{j_1}, \ldots, t_{j_k}\} \qquad \forall l \in 1..k | s_i \succ t_{j_l}}{\{s_1, \ldots, s_m\} \succ^{multi} \{t_1, \ldots, t_n\}}$$

multi-3: $$\frac{\{s_1, \ldots, s_m\} \setminus \{s_i\} \succ^{multi} \{t_1, \ldots, t_n\} \setminus \{t_j\} \qquad s_i = t_j}{\{s_1, \ldots, s_m\} \succ^{multi} \{t_1, \ldots, t_n\}}$$

Kort sagt er $M \succ^{multi} N$ dersom $M \neq N$ og N kan fås fra M ved å erstatte elementer i M med ingen eller flere elementer som er $\succ$-mindre. Ordningen $\succ^{multi}$ er velfundert over multiset til en mengde S hvis og bare hvis $\succ$ er velfundert over S.

Eksempel 2.5 Gitt de naturlige tallene og den vanlige partielle ordningen $>$ over disse. Da gjelder

$$M = \{3, 5, 5, 1, 0\} >^{multi} \{4, 5, 3, 3, 3, 2, 1, 1\} = N$$

fordi N kan fås fra M ved å erstatte én forekomst av 5 med $\{4, 3, 3, 2, 1\}$ og én 0 med ingenting.

Én av de vanligste forenklingsordningene er *multiset stiordning*, *mpo*, som kalles *recursive path ordering* i [Der82] og som med min notasjon defineres:

Definisjon 2.18 (Multiset stiordning som partiell ordning [Der82]) *La $\succ_F$ være en partiell ordning over en mengde funksjonssymboler F.* Multiset stiordningen $\succ_{mpo(\succ_F)}$ *indusert av $\succ_F$ er definert rekursivt over termer i $T(F)$ ved*

mpo-1: $$\frac{t_i \succeq_{mpo(\succ_F)} t'}{f(\ldots, t_i, \ldots) \succ_{mpo(\succ_F)} t'}$$

mpo-2: $$\frac{f \succ_F g \qquad \forall i \in 1..n | f(t_1, \ldots, t_m) \succ_{mpo(\succ_F)} t'_i}{f(t_1, \ldots, t_m) \succ_{mpo(\succ_F)} g(t'_1, \ldots, t'_n)}$$

mpo-3: $$\frac{\{t_1, \ldots, t_n\} \succ^{multi}_{mpo(\succ_F)} \{t'_1, \ldots, t'_n\}}{f(t_1, \ldots, t_n) \succ_{mpo(\succ_F)} f(t'_1, \ldots, t'_n)}$$

Jeg skriver $\succ_{mpo}$ for $\succ_{mpo(\succ_F)}$ når $\succ_F$ fremkommer av sammenhengen eller er uinteressant.

Testen $\forall i \in 1..n \mid f(t_1, \ldots, t_n) \succ_{mpo} t_i$ er ikke nødvendig i *mpo-3* fordi $\{t_1, \ldots, t_m\} \succ^{multi}_{mpo} \{t'_1, \ldots, t'_n\}$ sikrer at enhver term t'_i alltid har en term t_j slik at $t_j \succeq_{mpo} t'_i$. Vi skal senere se at $\succ_{mpo(\succ_F)}$ er velfundert når $\succ_F$ er det.

Eksempel 2.6 La $F = \{a, b, f, g\}$ og presedensen $\succ_F$ være gitt ved $f \succ_F g$ og $a \succ_F b$. Da gjelder $f(b, a, a) \succ_{mpo} g(a, f(a, b, b))$ som kan vises

$$\frac{f \succ_F g \qquad \dfrac{a = a}{f(b,a,a) \succ_{mpo} a} \qquad \dfrac{\dfrac{a = a \quad b = b \quad \dfrac{a \succ_F b}{a \succ_{mpo} b}}{\{b,a,a\} \succ^{multi}_{mpo} \{a,b,b\}}}{f(b,a,a) \succ_{mpo} f(a,b,b)}}{f(b,a,a) \succ_{mpo} g(a, f(a,b,b))}$$

Multiset stiordning kan brukes til å vise terminering etter teorem 2.16 fordi

Teorem 2.19 ([Der82]) *Multiset stiordningen definert over er en forenklingsordning.*

2.3.6 Leksikografisk stiordning

Sammenligningen av termsekvenser i stiordninger kan også foretas leksikografisk, noe som gir den såkalte *leksikografiske stiordningen.*

Definisjon 2.20 (Leksikografisk utvidelse av partiell ordning) *Gitt en mengde S og en partiell ordning $\succ$ over S defineres den leksikografiske utvidelsen $\succ^{lex}$ over sekvenser av S ved*

$$\langle s_1, \ldots, s_m \rangle \succ^{lex} \langle t_1, \ldots, t_n \rangle$$

hvis og bare hvis

$$\exists j \in 1..m \mid (j = n + 1 \vee s_j \succ t_j) \wedge (\forall i < j | s_i = t_i)$$

Eksempelvis er postene i en telefonkatalog ordnet av en leksikografisk ordning over tegnsekvenser indusert av en alfabetisk ordning over enkelttegn. Hvis $(S, \succ)$ er en velfundert partielt ordnet mengde så er $\succ^{lex}$ velfundert over mengden av S-sekvenser med lengde mindre eller lik et tall n. Den leksikografiske varianten av stiordninger kan da defineres (som i bl. a. [Kir92]):

Definisjon 2.21 (Leksikografisk stiordning som partiell ordning) *La $\succ_F$ være en partiell ordning over en mengde funksjonssymboler F. Den* leksikografiske stiordningen *$\succ_{lpo(\succ_F)}$ indusert av $\succ_F$ er definert rekursivt over termer i $T(F)$ ved*

lpo-1:
$$\frac{t_i \succeq_{lpo(\succ_F)} t'}{f(\ldots, t_i, \ldots) \succ_{lpo(\succ_F)} t'}$$

lpo-2:
$$\frac{f \succ_F g \qquad \forall i \in 1..n | f(t_1, \ldots, t_m) \succ_{lpo(\succ_F)} t'_i}{f(t_1, \ldots, t_m) \succ_{lpo(\succ_F)} g(t'_1, \ldots, t'_n)}$$

lpo-3:
$$\frac{\langle t_1, \ldots, t_n \rangle \succ^{lex}_{lpo(\succ_F)} \langle t'_1, \ldots, t'_n \rangle \qquad \forall i \in 2..n | f(t_1, \ldots, t_n) \succ_{lpo(\succ_F)} t'_i}{f(t_1, \ldots, t_n) \succ_{lpo(\succ_F)} f(t'_1, \ldots, t'_n)}$$

Jeg skriver på vanlig måte $\succ_{lpo}$ for $\succ_{lpo(\succ_F)}$ når $\succ_F$ fremkommer av sammenhengen eller er uinteressant.

I punkt 3 er det denne gangen nødvendig å vise $\forall i \in 2..n | f(\ldots) \succ_{lpo} t'_i$ fordi selv om $t_1 \succ_{lpo} t'_1$, så kunne en annen t'_i være veldig stor.

Eksempel (forts. av eks. 2.6) La F og presedensen $\succ_F$ være som i eksempel 2.6. I *lpo* gjelder imidlertid den omvendte $g(a, f(a, b, b)) \succ_{lpo} f(b, a, a)$ fordi

$$\dfrac{\dfrac{\dfrac{\dfrac{a \succ_F b}{a \succ_{lpo} b}}{\langle a, b, b \rangle \succ^{lex}_{lpo} \langle b, a, a \rangle} \qquad \dfrac{a = a}{f(a, b, b) \succ_{lpo} a}}{f(a, b, b) \succ_{lpo} f(b, a, a)}}{g(a, f(a, b, b)) \succ_{lpo} f(b, a, a)}$$

Det er lett å se at den leksikografiske stiordningen, *lpo*, innehar subtermegenskapen og er ikke-refleksiv, mens det er litt verre å vise at *lpo* er transitiv. Beviset for transitivitet går som tilsvarende bevis for transitivitet av *mpo* i [Der82], slik at *lpo* definert over er en forenklingsordning.

2.3.7 Stiordninger med variable

Ordningene *lpo* og *mpo* er bare definert for variabelfrie termer, noe som gir å vise

$$l\sigma \succ_{po} r\sigma$$

for uendelig mange grunnsubstitusjoner σ for enhver $l \longrightarrow r$ i en regelmengde R for å påvise at R terminerer. Denne situasjonen er selvfølgelig ikke holdbar og ordningene må løftes opp til å klare å behandle termer med variable slik at

$$l \succ_{po} r \Longrightarrow l\sigma \succ_{po} r\sigma$$

for alle grunnsubstitusjoner σ. Dette gjøres ved å se på regelvariable som konstanter i stiordningene, der en variabel x er ikke sammenlignbar med noe annet symbol i ordningen $\succ_F$. Dette er antatt implisitt i mange artikler, og gyldigheten av $l \succ_{po} r \Longrightarrow l\sigma \succ_{po} r\sigma$ kan vises ved induksjon på oppbygningen av beviset for $l \succ_{mpo} r$ (alternativt $l \succ_{lpo} r$).

Det er også mulig å kombinere *lpo* og *mpo* ut i fra den generelle definisjonen av stiordninger. For noen funksjonssymboler kan $\succ'_{po}$ være en multiset sammenligning, for andre en leksikografisk sammenligning og for atter andre kan $\succ'_{po}$ sammenligne leksikografisk i en annen rekkefølge. Eksempelet nedenfor illustrerer dette.

Eksempel 2.7 La $F = \{a, b, f, g\}$ og et terminerende omskrivningssystem R være gitt ved reglene

$$f(a,b) \longrightarrow f(b,a) \quad (1)$$
$$g(x,a) \longrightarrow g(b,x) \quad (2)$$
$$h(x,a,x) \longrightarrow h(a,b,x) \quad (3)$$

R viser for det første at *lpo* og *mpo* ikke er sammenlignbare i styrke fordi termene i regel 1 ikke er *mpo*-sammenlignbare samtidig som $a \succ_F b$ gir $f(a,b) \succ_{lpo} f(b,a)$. Det omvendte er tilfelle med regel 2. Hverken *lpo* eller *mpo* kan brukes på regel 3.

Det er mulig å vise at R terminerer ved å sette $\succ'_{po}$ lik $\succ^{lex}_{po}$ dersom hovedfunksjonssymbolet er f, og la $\succ'_{po}$ være $\succ^{multi}_{po}$ dersom hovedsymbolet er g og til sist la $\succ'_{po}$ være en leksikografisk sammenligning i rekkefølge 2. element, 1. element og 3. element når hovedsymbolet er en h.

Denne stiordningen gir da at alle reglene er $\succ_{po}$-minskende når $a \succ_F b$.

2.3.8 Ordningene som prosedyrer

Vi søkte etter prosedyrer som kan fastslå terminering og må følgelig vise at det alltid er beregnbart hvorvidt $t \succ_{mpo} t'$ og $t \succ_{lpo} t'$. For begge ordningene gjelder at en inferensregelapplikasjon (brukt "oppover") minsker totalstørrelsen av det største termparet, samtidig

som som vi ikke introduserer uendelige mange nye termpar som må vurderes. Det kan imidlertid dukke opp problemer fordi vi ikke vet nøyaktig *hvilken* regelapplikasjon som skal utføres til enhver tid, slik at metoden involverer prøving og feiling.

Ordningen *mpo* har den vakre egenskapen at vi ved ethvert tidspunkt vet nøyaktig hvilken av reglene *mpo-1, mpo-2* eller *mpo-3* som skal brukes, fordi *mpo-1* bare trenger å brukes dersom $f \not\succeq_F g$ for hovedsymboler f og g i termene t og t'. Valgets kvaler i *mpo* ligger i *mpo-3* der vi ikke vet hvilke subtermer som skal sammenlignes med hverandre. Heldigvis er det bare et endelig antall kombinasjoner som kan prøves mot hverandre.

Den leksikografiske stiordningen har ikke egenskapen at man hele tiden vet hvilken inferensregel som skal velges. Gitt $a \succ_F b$ kan ikke regel *lpo-2* velges for å vise

$$f(b, f(a, b)) \succ_{lpo} f(a, b)$$

Ordningen *lpo* involverer altså prøving og feiling med hensyn til hvilke inferensregler man skal bruke, men *lpo-3* slipper å finne ut hvilke subtermer som skal sammenlignes, men har i tillegg den ulempen at at alle subtermer i t' må sammenlignes med t. Siden denne testen ligger implisitt i *mpo-3* vil jeg likevel si at *lpo* er en anelse raskere, og brukes kanskje derfor mest i implementasjoner, mens *mpo* kan sies å være mer elegant. Ordningene er som vist heller ikke sammenlignbare i styrke.

Stiordningene har videre den fordelen at man bare trenger å definere et forhold mellom funksjonssymboler, noe som gjør dem veldig implementeringsvennlige. Til sist har de den nyttige egenskapen at de er inkrementérbare. Det betyr at presedensen $\succ_F$ kan utvides underveis, slik at $l \succ_{po(\succ_F)} r$ medfører $l \succ_{po(\succ'_F)} r$ dersom $\succ'_F$ er en utvidelse av $\succ_F$ (dvs. $f \succ_F g$ impliserer $f \succ_{F'} g$ for alle $f, g \in F$). Dette gjør at man bare trenger å utvide presedensen mellom funksjonssymbolene etter hvert som "foreløpig ikke-orienterbare" likninger dukker opp f. eks. i en kompletterings-prosedyre uten at man ødelegger tidligere viste resultater.

Til slutt kan det bemerkes at stiordninger generelt ikke er velfunderte. Gitt en uendelig mengde konstanter $F = \{a_1, a_2, a_3, \ldots\}$ og en presedens $a_1 \succ_F a_2 \succ_F \cdots$ fins det uendelig sekvens $a_1 \succ_{po} a_2 \succ_{po} a_3 \succ_{po} \cdots$ ved regel 2. Stiordningene er imidlertid velfunderte for avledninger, fordi en *endelig* regelmengde R ikke kan gi opphav til en uendelig omskrivningssekvens $a_1 \xrightarrow[R]{} a_2 \xrightarrow[R]{} a_3 \xrightarrow[R]{} \cdots$.

2.3.9 Andre forenklingsordninger

Polynomiske og eksponensielle ordninger

Polynomiske ordninger tilordner enhver grunnterm en positiv heltallig vekt ved at ethvert n-ært funksjonssymbol f blir tilordnet et heltallspolynom $F(x_1, \ldots, x_n)$. Koeffisientene må velges slik at monotoniegenskapen og subtermegenskapen opprettholdes, noe som eksempelvis kan oppnås med å bare bruke positive koeffisienter i polynomet.

Dersom $vekt(l\sigma) > vekt(r\sigma)$ for alle regler $l \longrightarrow r$ i et omskrivningssystem R så innebærer $t \xrightarrow[R]{} t'$ at $vekt(t) > vekt(t')$. Siden enhver grunnterm må ha ikke-negativ heltallig vekt kan det ikke finnes en uendelig omskrivningssekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} t_3 \xrightarrow[R]{} \cdots$$

Eksempel 2.8 La F bestå av bl. a. funksjonssymbolene a, f og g med aritet hhv. 0, 2 og 1. Anta at det er gitt en regel

$$g(f(x,y)) \longrightarrow f(g(x),y)$$

Tilordner man a, f og g hhv. polynomene $A = 1$, $F(x,y) = 2x + y$ og $G(x) = 2x$ er $vekt(g(f(x,y))\sigma) = 4 \cdot vekt(x\sigma) + 2 \cdot vekt(y\sigma)$ alltid større enn $vekt(f(g(x),y)\sigma) = 4 \cdot vekt(x\sigma) + vekt(y\sigma)$ fordi enhver grunnterm alltid har positiv vekt slik at $2 \cdot vekt(y\sigma)$ er større enn $vekt(y\sigma)$ for enhver grunnterm $y\sigma$. Systemet er derfor terminerende.

Polynomiske ordninger tilordner en vekt i det velfunderte domenet $\mathcal{N}$ til hver term. Forenklingsordningsbegrepet gjør at man like gjerne kan tilordne reelle tall til en term. Selv om $>$ ikke er velfundert over mengden $\mathcal{R}$ av alle reelle tall holder det (fordi $>$ over $\mathcal{R}$ er transitiv og ikke-refleksiv) å vise at vektleggingen er monoton, dvs.

$$x_i > x'_i \Longrightarrow F(\ldots, x_i, \ldots) > F(\ldots, x'_i, \ldots)$$

og innehar subtermegenskapen

$$F(\ldots, x_i, \ldots) > x_i$$

for alle reelle tall x_i, x'_i. Ordningen $\succ$ indusert av vektleggingen ved $t \succ t' \Longleftrightarrow vekt(t) > vekt(t')$ er da en forenklingsordning og kan brukes til å vise terminering av endelige systemer. Funksjonen F trenger naturligvis ikke heller være et polynom, men kan være en vilkårlig eksponensiell funksjon som oppfyller kravene over. På denne måten får vi *eksponensielle ordninger*.

Polynomiske ordninger er ikke så implementeringsvennlige, fordi det ikke er så lett å definere de nødvendige polynomene. De er heller ikke inkrementérbare i samme utstrekning som stiordningene, noe som medfører at de egner seg dårligere i kompletteringsprosedyrer. Jeg vil derfor ikke se mer på slike ordninger.

Andre semantiske ordninger

Av andre rene semantiske ordninger vi har sett på tidligere, kan jeg nevne eksempelvis

$$t \succ t' \Longleftrightarrow \text{antall symboler i } t \ > \ \text{antall symboler i } t'$$

og hvorfor ikke

$$t \succ t' \Longleftrightarrow (\text{antall } f \text{ i } t - \text{ antall } g \text{ i } t\ ) \ > \ (\text{antall } f \text{ i } t' - \text{ antall } g \text{ i } t')$$

Semantiske ordninger er gjerne spesialkonstruerte utifra formen til visse termer i reglene og kan derfor ikke brukes som generelle termineringsordninger. De brukes ofte for teoretisk å påvise terminering av "vanskelige" systemer. Jeg vil derfor ikke si noe særlig mer om disse; mange eksempler på semantiske ordninger fins i [Der87].

2.4 Kvasiordninger og kvasiforenklingsordninger

Kvasiordninger både over funksjonssymboler og som termineringsordninger over termer, som tillater ekvivalens mellom funksjonssymboler, vil spille en stor rolle for å oppnå brukbar styrke i mine ordninger for typeordnet terminering.

Jeg vil derfor redefinere *mpo* og *lpo* i dette avsnittet slik at de blir kvasiordninger.

2.4.1 Kvasiordninger

Ordningene *mpo* og *lpo* definert tidligere bygger på en ikke-refleksiv presedens $\succ_F$ mellom funksjonssymboler. Følgende eksempel avdekker en stor svakhet ved dette og viser at det kan være nødvendig å la to funksjonssymboler f og g ha lik presedens i ordningene for å kunne påvise terminering.

Eksempel 2.9 La $F = \{a, b, f, g\}$ og et omskriningssystem R være gitt ved regelmengden

$$\begin{array}{ll} f(a) \longrightarrow g(b) & (1) \\ g(a) \longrightarrow f(b) & (2) \\ f(x) \longrightarrow a & (3) \end{array}$$

Det er lett å se at R terminerer, men dette kan ikke vises med *mpo* (eller *lpo* som er lik *mpo* for systemer med bare nullære og unære funksjonssymboler), fordi regel 3 forlanger $f \succ_F a$ av presedensen $\succ_F$. For å vise $g(a) \succ_{mpo} f(b)$ må da $g \succ_F f$. Med presedensen $g \succ_F f \succ_F a$ er det umulig å vise $f(a) \succ_{mpo} g(b)$.

Dersom man lar f og g være "like" ved f. eks. erstatte enhver forekomst av g med en f, får man et i termineringshenseende "svakere" system siden enhver uendelig sekvens

$$t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots$$

i det opprinnelige systemet også vil gi en uendelig sekvens

$$t'_1 \xrightarrow[R]{} t'_2 \xrightarrow[R]{} \cdots$$

når alle g'er er erstattet med f'er i termene og i reglene i omskrivningssystemet.

Imidlertid er det greit å vise terminering av det svakere systemet

$$\begin{array}{ll} f(a) \longrightarrow f(b) & (1) \\ f(a) \longrightarrow f(b) & (2) \\ f(x) \longrightarrow a & (3) \end{array}$$

med presedensen $f \succ_F a \succ_F b$.

Eksempelet antyder at det kan være ønskelig å definere en presedens som tillater "ekvivalens" mellom funksjonssymboler. For ikke å ødelegge transitive egenskaper må man imidlertid sikre at $f \succ_F g$ dersom g og h er "ekvivalente" og $f \succ_F h$. En *kvasiordning* er en kraftig og elegant form for relasjon som sikrer dette.

Definisjon 2.22 (Kvasiordning) *En kvasiordning over en mengde S er en transitiv og refleksiv binær relasjon $\succsim$ over elementer i S.*

For kvasiordninger skriver jeg også $a \precsim b$ for $b \succsim a$.

En kvasiordning $\succsim$ induserer relasjonen $\approx$ ved $a \approx b$ hvis og bare hvis $a \succsim b$ og $b \succsim a$, og relasjonen $\succ$ ved $\succsim$ men ikke $\precsim$. Den strikte delen $\succ$ av en kvasiordning $\succsim$ sies å være *velfundert* hvis og bare hvis enhver uendelig kvasiminskende sekvens

$$s_1 \succsim s_2 \succsim \cdots \succsim s_j \succsim \cdots \succsim s_k \succsim \cdots$$

av elementer i S inneholder et par av elementer $s_j \precsim s_k$ der $j < k$. En kvasiordning $\succsim$ kalles velfundert når den strikte delen er velfundert.

Eksempel 2.10 La kvasiordningen $\succsim$ være definert over par av naturlige tall ved $\langle n_1, n_2\rangle \succsim \langle n_3, n_4\rangle \iff n_1+n_2 \geq n_3+n_4$. Da er $\langle 2,3\rangle \approx \langle 1,4\rangle$ og $\langle 2,3\rangle \succ \langle 1,3\rangle$ og $\succsim$ er velfundert.

Jeg vil senere få mye bruk for følgende lemma:

Lemma 2.23 *Gitt en kvasiordning $\succsim$ over en mengde S. Relasjonen $\approx$ definert ved $\approx = \succsim \cap \precsim$ er en ekvivalensrelasjon over S.*

Bevis: Må vise at $\approx$ er refleksiv, symmetrisk og transitiv. Refleksivitet følger av at en kvasiordning per definisjon er refleksiv, slik at $a \succsim a$ (og selvfølgelig også $a \precsim a$) gjelder for alle $a \in S$. Anta så at $a \approx b$, som per definisjon betyr $a \succsim b$ og $b \succsim a$. Den symmetriske $b \approx a$ følger fra disse to sistnevnte ulikhetene. Anta til sist at $a \approx b$ og $b \approx c$, som kommer av $a \succsim b$, $b \succsim a$, $b \succsim c$ og $c \succsim b$. Da følger $a \approx c$ av at $\succsim$ er transitiv slik at $a \succsim c$ og $c \succsim a$. □

Fra nå av vil presedensen over funksjonssymboler som brukes i stiordningene være en kvasiordning $\succsim_F$. Meningen er å betrakte $\approx_F$-like funksjonssymboler som like under termineringsordningene, så den første forandringen man kunne tenke på er å erstatte den siste regelen i definisjonen av stiordninger med

$$\frac{f \approx_F g \qquad \langle t_1, \ldots, t_m\rangle \succ'_{po} \langle t'_1, \ldots, t'_n\rangle \qquad \forall i \in 1..n \mid f(t_1, \ldots, t_m) \succ_{po} t'_i}{f(t_1, \ldots, t_m) \succ_{po} g(t'_1, \ldots, t'_n)}$$

2.4.2 Kvasiforenklingsordninger

For at $\approx_F$-like funksjonsymboler skal betraktes som ekvivalente av ordningen, må stiordningene forandres mer enn antydet.

Eksempel 2.11 Anta gitt $F = \{a, b, c, d, f, g\}$ og presedensen $a \approx_F b$ og $c \succ_F d$. Dersom a og b skal sees på som like burde man kunne vise $f(g(a), c) \succ_{lpo} f(g(b), d)$, noe som ikke er mulig å vise med *lpo* selv med forandringen nevnt over. Grunnen til det er at termene $g(a)$ og $g(b)$ skulle vært "*lpo*-like", men et slikt begrep har vi foreløpig ikke i denne oppgaven. Defineres *lpo* som en kvasiordning, kan man ha håp om å oppnå et "*lpo*-likhets-begrep".

I lys av eksempelet over gjøres termineringsordningene om til kvasiordninger over termer.

Definisjon 2.24 (Monotoni) *En kvasiordning $\succsim$ over en mengde termer $T(F)$ er* monoton *hvis og bare hvis*

$$t \succsim t' \Longrightarrow f(\ldots, t, \ldots) \succsim f(\ldots, t', \ldots)$$

for alle termer t, t' i $T(F)$ og funksjonssymboler f i F.

Relasjonen $\approx$ er en kongruensrelasjon når $\succsim$ er monoton. Definisjonen forlanger imidlertid *ikke* $t \succ t' \Longrightarrow f(\ldots, t, \ldots) \succ f(\ldots, t', \ldots)$, noe som gjør bevisene for teoremene nedenfor litt mer kompliserte.

På samme måte som for som for partielle termineringsordninger finnes også forenklingsordninger for kvasiordninger.

Definisjon 2.25 (Kvasiforenklingsordning) *En monoton kvasiordning* $\succsim$ *er en* kvasiforenklingsordning *over en mengde termer* $T(F)$ *dersom den innehar subtermegenskapen*

$$f(\ldots, t, \ldots) \succsim t$$

for alle termer i $T(F)$.

En kvasiforenklingsordning $\succsim$ er en utvidelse av $\trianglerighteq$ og den strikte delen $\succ$ er da velfundert for avledninger.

Teorem 2.26 ([Der87]) *Et* endelig *omskrivningssystem* R *terminerer for en mengde* $T(F)$ *dersom det fins en kvasiforenklingsordning* $\succsim$ *over* $T(F)$ *slik at*

$$l\sigma \succ r\sigma$$

for alle substitusjoner σ *av alle regler* $l \longrightarrow r$ *i* R.

For å vise at en ordning $\succsim$ kan brukes til å fastslå terminering, er det altså tilstrekkelig å vise at $\succsim$ er refleksiv[6], transitiv, monoton og innehar subtermegenskapen.

Problemet med det ovenstående er at man må vise $l\sigma \succ r\sigma$ for alle regelsubstitusjoner samtidig som ordningen er definert ved $\succsim$. Man må altså ha $l\sigma \succsim r\sigma$ men *ikke* $r\sigma \succsim l\sigma$. I praksis blir dette ofte gjort ved at man karakteriserer relasjonen $\approx$ og først fastslår at $l\sigma$ og $r\sigma$ ikke er $\approx$-like og deretter viser $l\sigma \succsim r\sigma$.

Før *mpo* og *lpo* redefineres, må multiset og leksikografisk utvidelse av en kvasiordning defineres.

Definisjon 2.27 (Leksikografisk utvidelse av en kvasiordning) *Dersom* $\succsim$ *er en kvasiordning over en mengde* S*, defineres den leksikografiske utvidelsen* $\succsim^{lex}$ *over endelige* S*-sekvenser ved*

$$\langle s_1, \ldots, s_m \rangle \succsim^{lex} \langle s'_1, \ldots, s'_n \rangle$$

hvis og bare hvis enten

$$m = n \wedge \forall i \in 1..m | s_i \succsim s'_i$$

eller

$$\exists j \in 1..m \mid (j = n + 1 \vee s_j \succ s'_j) \wedge (\forall i < j | s_i \succsim s'_i)$$

for alle elementer i S.

Dette er muligens en "løsere" definisjon enn man ellers ser fordi jeg har byttet ut $\approx$ med $\succsim$ for å spare noen unødvendige beregninger, men $\succsim$ fanger inn de tilfellene med $\approx$ i de to betingelsene.

Definisjon 2.28 (Multiset-utvidelse av en kvasiordning) *Gitt en mengde* S *og en kvasiordning* $\succsim$ *over* S *defineres multiset-utvidelsen* $\succsim^{multi}$ *av* $\succsim$ *over endelige* S*-multiset rekursivt ved*

[6] Hvis $\succsim$ skal brukes som teoremet over indikerer er det strengt tatt ikke engang nødvendig å vise at $\succsim$ er refleksiv!

multi-1: $$\emptyset \succsim^{multi} \emptyset$$

multi-2: $$\frac{\{s_1,\ldots,s_m\} \setminus \{s_i\} \succsim^{multi} \{t_1,\ldots,t_n\} \setminus \{t_{j_1},\ldots,t_{j_k}\} \quad (k \geq 0) \quad \forall l \in 1..k | s_i \succ t_{j_l}}{\{s_1,\ldots,s_m\} \succsim^{multi} \{t_1,\ldots,t_n\}}$$

multi-3: $$\frac{\{s_1,\ldots,s_m\} \setminus \{s_i\} \succsim^{multi} \{t_1,\ldots,t_n\} \setminus \{t_j\} \qquad s_i \approx t_j}{\{s_1,\ldots,s_m\} \succsim^{multi} \{t_1,\ldots,t_n\}}$$

for alle elementer $s_1,\ldots,t_1,\ldots$ *i* S.

Utvidelsene bygger på samme prinsipper som de tilsvarende utvidelsene for partielle ordninger. Relasjonen $\succsim^{multi}$ over multiset til S er velfundert hvis og bare hvis kvasiordningen $\succsim$ er velfundert over S.

Det følger fra multiset-definisjonen at

$$\{s_1,\ldots,s_m\} \approx^{multi} \{s'_1,\ldots,s'_n\}$$

hvis og bare hvis $m = n$ og det fins en permutasjon π av $1\ldots m$ slik at for alle $i \in 1..m$ er $s_i \approx s'_{\pi(i)}$.

2.4.3 Leksikografisk stiordning

Den leksikografiske stiordningen kan nå redefineres til å bli en kvasiordning over termer.

Definisjon 2.29 (Kvasiordningen leksikografisk stiordning) *La* $\succsim_F$ *være en kvasiordning over en mengde funksjonssymboler* F. *Den* leksikografiske stiordningen $\succsim_{lpo(\succsim_F)}$ *indusert av* $\succsim_F$ *er definert rekursivt over termer i* $T(F)$ *ved*

lpo-1: $$\frac{t_i \succsim_{lpo(\succsim_F)} t'}{f(\ldots,t_i,\ldots) \succsim_{lpo(\succsim_F)} t'}$$

lpo-2: $$\frac{f \succ_F g \qquad \forall i \in 1..n | f(t_1,\ldots,t_m) \succ_{lpo(\succsim_F)} t'_i}{f(t_1,\ldots,t_m) \succsim_{lpo(\succsim_F)} g(t'_1,\ldots,t'_n)}$$

lpo-3: $$\frac{f \approx_F g \quad \langle t_1,\ldots,t_m\rangle \succsim^{lex}_{lpo(\succsim_F)} \langle t'_1,\ldots,t'_n\rangle \quad \forall i \in 2..n | f(t_1,\ldots,t_m) \succ_{lpo(\succsim_F)} t'_i}{f(t_1,\ldots,t_m) \succsim_{lpo(\succsim_F)} g(t'_1,\ldots,t'_n)}$$

Jeg skriver på vanlig måte $\succsim_{lpo}$ *for* $\succsim_{lpo(\succsim_F)}$ *når* $\succsim_F$ *fremkommer av sammenhengen eller er uinteressant.*

Definisjonen inneholder ingen overraskelser, slik at $\succsim_{lpo}$ er den opplagte utvidelsen av $\succ_{lpo}$ fra definisjon 2.21 til en kvasiordning. Ordningen *lpo* definert over er en kvasiordning [Der87] og to termer er $\approx_{lpo}$-like dersom den ene kan fås fra den andre ved å erstatte funksjonssymboler med $\approx_F$-like symboler.

Fra nå av vil $\succ_{lpo}$ stå for den strikte delen av $\succsim_{lpo}$, og ikke for relasjonen $\succ_{lpo}$ definert i avsnitt 2.3. Denne sistnevnte relasjonen vil jeg skrive $\succ_{gammel\ lpo}$ i tilfelle jeg vil ha bruk for den.

Ordningen $\succsim_{lpo}$ kan brukes til å vise terminering fordi

Teorem 2.30 (Kamin og Lévy, sitert i [Der87]) *Den strikte delen* $\succ_{lpo(\succsim_F)}$ *av den leksikografiske stiordningen* $\succsim_{lpo(\succsim_F)}$ *indusert av en kvasiordning* $\succsim_F$ *over en mengde funksjonssymboler* F *er en forenklingsordning over mengden* $T(F)$ *av termer.*

Eksempel (forts. av eks. 2.11) Gitt $F = \{a, b, c, d, f, g\}$ og presedensen $a \approx_F b$ og $c \succ_F d$. I 2.11 viste det seg at i det første forsøket på å behandle "ekvivalente" funksjonssymboler så gjelder ikke $f(g(a), c) \succ_{gammel\ lpo} f(g(b), d)$. Med den nye definisjonen kan man imidlertid vise $f(g(a), c) \succsim_{lpo} f(g(b), d)$ på følgende måte

$$\dfrac{\dfrac{g \approx_F g \qquad \dfrac{\dfrac{\dfrac{a \approx_F b}{a \succsim_{lpo} b}}{a \succsim_{lpo}^{lex} b}}{g(a) \succsim_{lpo} g(b)} \qquad \dfrac{c \succ_F d}{c \succsim_{lpo} d}}{f \approx_F f \qquad \langle g(a), c\rangle \succsim_{lpo}^{lex} \langle g(b), d\rangle} \qquad \dfrac{c \succ_F d}{c \succsim_{lpo} d}}{f(g(a), c) \succsim_{lpo} f(g(b), d)} \qquad \dfrac{}{f(g(a), c) \succ_{lpo} d}$$

samtidig som termene ikke er $\approx_{lpo}$-like, slik at

$$f(g(a), c) \succ_{lpo} f(g(b), d)$$

gjelder.

Når (nye) $\succ_{lpo}$ er en forenklingsordning fører teorem 2.16 til at man kan vise terminering av endelige systemer ved å fastslå $l\sigma \succ_{lpo} r\sigma$ for alle regelsubstitusjoner i R.

På samme måte som tidligere er $\succ_{lpo}$ heller ikke denne gangen velfundert, men bare velfundert for avledninger. Ordningen *lpo* kan imidlertid også brukes til å vise terminering av uendelige store regelmengder som følge av neste teorem.

Teorem 2.31 (Kamin og Lévy, sitert i [Der87]) *En kvasiordning* $\succsim_F$ *over en mengde funksjonsymboler* F *er velfundert hvis og bare hvis den induserte leksikografiske stiordningen* lpo *over* $T(F)$ *er velfundert, sålenge ekvivalente funksjonssymboler har lik aritet.*

Nytten av *lpo* for uendelige regelmengder er dessverre begrenset av at ekvivalente funksjonssymboler må ha lik aritet.

2.4.4 Multiset stiordning om igjen

Den nye kvasiordningen multiset stiordning kan da defineres på denne måten:

Definisjon 2.32 ("Ny" multiset stiordning) *La* $\succsim_F$ *være en kvasiordning over en mengde funksjonssymboler* F. Multiset stiordningen $\succsim_{mpo(\succsim_F)}$ *indusert av* $\succsim_F$ *er definert rekursivt over termer i* $T(F)$ *ved*

mpo-1: $$\dfrac{t_i \succsim_{mpo(\succsim_F)} t'}{f(\ldots, t_i, \ldots) \succsim_{mpo(\succsim_F)} t'}$$

mpo-2: $$\dfrac{f \succ_F g \qquad \forall i \in 1..n | f(t_1, \ldots, t_m) \succ_{mpo(\succsim_F)} t'_i}{f(t_1, \ldots, t_m) \succsim_{mpo(\succsim_F)} g(t'_1, \ldots, t'_n)}$$

mpo-3: $$\dfrac{f \approx_F g \qquad \{t_1, \ldots, t_m\} \succsim_{mpo(\succsim_F)}^{multi} \{t'_1, \ldots, t'_n\}}{f(t_1, \ldots, t_m) \succsim_{mpo(\succsim_F)} g(t'_1, \ldots, t'_n)}$$

Jeg skriver $\succsim_{mpo}$ *for* $\succsim_{mpo(\succsim_F)}$ *når* $\succsim_F$ *fremkommer av sammenhengen eller er uinteressant.*

Heller ikke denne definisjonen byr på noen overraskelser i forhold til den tidligere definisjonen av $\succ_{mpo}$ i avsnitt 2.3. Det lar seg vise at *mpo* er en kvasiforenklingsordning. To termer er $\approx_{mpo}$-ekvivalente hvis den ene kan fås fra den andre ved å erstatte funksjonssymboler med $\approx_F$-ekvivalente symboler og ved (rekursivt) å permutere subtermene, dvs. $f(t_1, \ldots, t_m) \approx_{mpo} g(t'_1, \ldots, t'_n)$ hvis $m = n$, $f \approx_F g$ og det fins en permutasjon π av $1 \ldots m$ slik at $t_i \approx_{mpo} t'_{\pi(i)}$ for alle i fra 1 til m. Videre i oppgaven vil $\succ_{mpo}$ stå for den strikte delen av kvasiordningen $\succsim_{mpo}$, dvs. for relasjonen $\succsim_{mpo} \setminus \precsim_{mpo}$.

Teorem 2.33 ([Der87]) *For enhver kvasiordning* $\succsim_F$ *over en mengde funksjonssymboler* F *er multiset stiordningen* $\succ_{mpo(\succsim_F)}$ *(den strikte delen av* $\succsim_{mpo(\succsim_F)}$ *) en forenklingsordning over* $T(F)$.

Sammen med teorem 2.16 gir dette at $\succ_{mpo}$ er velfundert for avledninger for endelige regelmengder R. Skulle R være uendelig kan man ty til følgende teorem:

Teorem 2.34 ([Der87]) *En kvasiordning* $\succsim_F$ *over en mengde funksjonssymboler* F *er velfundert hvis og bare hvis den induserte kvasiordningen* $\succsim_{mpo(\succsim_F)}$ *over termer i* $T(F)$ *er velfundert.*

Dette betyr at dersom enten R eller F er endelig (eller i verste fall at $\succsim_F$ er velfundert), og man kan vise

$$l\sigma \succ_{mpo} r\sigma$$

for alle substitusjoner σ for alle regler $l \longrightarrow r$ i R så terminerer R.

Det kan være interessant å merke seg at forskjellen mellom "gamle" *mpo* definert i avsnitt 2.3 og kvasiordningsvarianten er større enn at $\approx_F$-like funksjonssymboler får samme "verdi" innenfor ordningen.

Eksempel 2.12 La $F = \{a, b, f, g\}$ og presedensen $\succsim_F$ være gitt ved (den opplagte utvidelsen av) $a \succ_F b \succ_F g \succ_F f$. Ingen (forskjellige) funksjonssymboler er ekvivalente i $\succsim_F$.

Med "gamle" *mpo* gjelder ikke

$$f(g(a,b),a) \succ_{gammel\ mpo} f(g(b,a),b)$$

fordi $\{g(a,b),a\}$ ikke kan sammenlignes med $\{g(b,a),b\}$ under $\succ^{multi}_{gammel\ \ mpo}$ fordi det ikke er noe begrep for $\approx_{gammel\ mpo}$ over termer utover identitet.

$$f(g(a,b),a) \succ_{mpo} f(g(b,a),b)$$

gjelder i "nye" *mpo* og kan vises med følgende utledning:

$$\dfrac{\dfrac{g \approx_F g \qquad \dfrac{\dfrac{a \approx_F a}{a \succsim_{mpo} a} \qquad \dfrac{\dfrac{b \approx_F b}{b \succsim_{mpo} b} \qquad a \succ_F b}{a \succsim_{mpo} b}}{\{a,b\} \succsim^{multi}_{mpo} \{b,a\}}}{g(a,b) \succsim_{mpo} g(b,a)} \qquad f \approx_F f}{\{g(a,b),a\} \succsim^{multi}_{mpo} \{g(b,a),b\}}}{f(g(a,b),a) \succsim_{mpo} f(g(b,a),b)}$$

Legg merke til at for å vise $\{g(a,b),a\} \succsim_{mpo}^{multi} \{g(b,a),b\}$ er det ikke nødvendig å vise både $g(a,b) \succsim_{mpo} g(b,a)$ *og* $g(b,a) \succsim_{mpo} g(a,b)$. Siden b er mindre enn a er heller ikke $f(g(a,b),a)$ og $f(g(b,a),b)$ $\approx_{mpo}$-like slik at man dermed kan slutte ønskede $f(g(a,b),a) \succ_{mpo} f(g(b,a),b)$.

Ordningene har ellers de samme særtrekkene og beregnbarhetsegenskapene som ble beskrevet i avsnitt 2.3 og kan brukes som prosedyrer for å fastslå terminering. Det eneste man må huske på er å vise at to termer i en regel ikke er $\approx$-like når man har vist at regelen er $\succsim$-minskende, og dette er en triviell sak.

2.4.5 Utvidelser av ordningene

De to kvasiforenklingsordningene kan også kombineres slik at noen sammenligninger over termtupler foretas med multiset-sammenligninger, andre med leksikografiske sammenligninger for $\approx_F$-like funksjonssymboler.

På samme måte som tidligere kan ordningene løftes til å behandle regler med variable slik at

$$l \succ_{po} r \Longrightarrow l\sigma \succ_{po} r\sigma$$

for kvasiordningene $\succsim_{mpo}$ og $\succsim_{lpo}$ ved at variable også denne gangen er usammenlignbare i $\succsim_F$ med andre symboler enn seg selv. Jeg vil implisitt anta dette i resten av oppgaven.

To velfunderte kvasiordninger $\succsim$ og $\succsim'$ over termer kan kombineres leksikografisk til en ny velfundert kvasiordning $\succsim''$ ved at

$$l \succsim'' r \Longleftrightarrow l \succ r \vee (l \approx r \wedge l \succsim' r)$$

På denne måten kan man kombinere mange termineringsordninger (eller samme ordning med forskjellige presedenser!) ved at $\succsim''$ er en velfundert kvasiordning som kan kombineres med $\succsim'''$ til en ny velfundert ordning $\succsim''''$ og så videre.

2.4.6 Notasjonen videre

Kvasiforenklingsordningene er som vi sett ekte sterkere enn de "opprinnelige" $\succ_{lpo}$ og $\succ_{mpo}$. I implementasjoner i dag er det (ifølge [Pla93]) kvasiforenklingsordninger som brukes. Når jeg i senere kapitler nevner presedenser og ordninger er det altså hele tiden kvasiordningsvariantene jeg omtaler.

Kapittel 3

Typeordnet algebra

I de fleste av dagens programmeringsspråk er funksjoner og objekter gitt en *type*. Typingen gjør programmeringen klarere og kan gjøre det enklere å finne feil før og under kjøring. Typing abstraherer også bort det maskinavhengige og lar oss arbeide med heltall, tekster og klasseobjekter i stedet for bitsekvenser. I språk med *sterk typing*, som kan formaliseres av *mangetypet algebra*, er ethvert objekt gitt én bestemt type samtidig som enhver funksjon tar en sekvens av argumenter av gitte typer og gir et resultat av én bestemt type.

Sterk typing er imidlertid litt restriktivt og mangler fleksibilitet. Det er ofte mer hensiktsmessig å la noen typer være inneholdt i andre typer og la disse "subtypene" arve egenskapene til "supertypen". En slik inndeling speiler også virkeligheten bedre, da begreper gjerne er inneholdt i andre begreper. Eksempelvis er både aper og fugler dyr, og alle orangutanger[1] er aper. Alle aper og fugler arver egenskapene til dyr generelt (som "må spise"), men har også egenskaper som ikke alle dyr har (hhv. "har fingre" og "har vinger"). Vi har *arv* gjennom at orangutanger arver egenskapene til aper, og dermed også egenskapene til alle dyr, samtidig som disse har spesialtrekk, som f. eks. "ensom*ulv*" og "har lange underarmer".

Hierarkiet kan passelig modelleres av en partiell ordning $<$ slik at $\mathsf{orangutang} < \mathsf{ape} < \mathsf{dyr}$ og $\mathsf{fugl} < \mathsf{dyr}$, der vi sier at a er en *subtype* til b, og at b er en *supertype* til a dersom $a < b$. Ser man på forekomstene til typene/begrepene finner man naturlig nok at $<$ representerer delmengderelasjonen fordi "alle forekomster av orangutanger" $\subseteq$ "alle forekomster av aper" $\subseteq$ "alle forekomster av dyr". Subtypebegrepet med evnen til nedarving av egenskaper ble utviklet i programmeringsspråket Simula. *Typeordnet algebra* (eng. "order-sorted algebra") formaliserer disse begrepene og idéene.

Det er ofte nødvendig å definere noen funksjoner i modellen. Funksjonen "alder" brukes på alle dyr, og må følgelig ta en ape, eller t. o. m. en orangutang som argument. Andre funksjoner, som f. eks. "vingespenn" er partielle, dvs. de gir ikke mening for andre dyr enn fugler. Ser vi på funksjonen alder fins det dyr, f. eks. skilpadder, som kan bli over 200 år gamle, mens empiri viser at ingen orangutang blir over 60 år. Dette faktum kan utnyttes slik at funksjonen *alder* kan ha profilene

$$alder : \mathsf{dyr} \rightarrow \{0 \ldots 250\}$$
$$alder : \mathsf{orangutang} \rightarrow \{0 \ldots 60\}$$

[1] som det dessverre er altfor få igjen av

når $\{0 \ldots 250\}$ og $\{0 \ldots 60\}$ er to Pascal-aktige intervalltyper der typen $\{0 \ldots 60\}$ er en subtype av $\{0 \ldots 250\}$. *Subtypepolymorfi*-egenskapen i typeordnet algebra gjør at $alder(ole)$ har éntydig verdi i alle dyr-algebraer også når ole er en orangutang.

Hva vinner man med typeordnet algebra? Ved siden av at det som vi allerede har sett ofte er naturlig å dele inn verden i "subtyper", er kanskje den største styrken til til typeordnet algebra at den tillater oss å håndtere partielle funksjoner og ikke-veldefinerte uttrykk meget elegant. To klassiske tilfeller er (heltalls-) divisjon med 0 og forsøk på å fjerne top-elementet på en tom stakk. Det har tidligere vært vanskelig å gi algebraiske spesifikasjoner av abstrakte datatyper som inneholder partielle funksjoner. I typeordnet algebra løses slike problemer enkelt ved å innføre hhv. subtypene $NzNat < Nat$, der $NzNat$ står for alle ekte positive heltall, og $NeStack < Stack$ som representerer ikke-tomme stakker. Divisjonsoperatoren kan deklareres

$$/ : Nat, NzNat \rightarrow Nat$$

Følgende eksempel viser hvordan datatypen $NatStack$ kan spesifiseres i typeordnet algebra:

Eksempel 3.1 Gitt typen Nat kan spesifikasjonen $NatStack$ være

Typer:	$Nat, Stack, NeStack$
	$NeStack < Stack$
Funksjoner:	$empty : Stack$
	$push : Stack, Nat \rightarrow NeStack$
	$pop : NeStack \rightarrow Stack$
	$top : NeStack \rightarrow Nat$
Likninger:	$(\forall s{:}Stack,\ n{:}Nat)\ \ pop(push(s,n)) \approx s$
	$(\forall s{:}Stack,\ n{:}Nat)\ \ top(push(s,n)) \approx n$

I tillegg til at det ofte er hensiktsmessig å arbeide med typeordnet algebra, er det av og til direkte nødvendig å innføre subtyper for å løse noen problemer. Et vanlig eksempel er problemet med å definere en *én-én-tydig generatorbasis* for alle heltall, dvs. en mengde funksjonssymboler der enhver irredusibel term bygget opp av disse symbolene representerer ett og bare ett heltall samtidig som alle heltallene blir representert av én eller annen slik term. Det synes meget vanskelig å definere en slik generatorbasis i utypet eller mangetypet algebra (se [Dahl92] for for en diskusjon rundt dette), men følgende typeordnede signatur gjør jobben:

Typer:	$Int, Zero, Nat, NzNat, Neg, NzNeg$
	$Zero < Nat < Int$
	$Zero < Neg < Int$
	$NzNat < Nat$
	$NzNeg < Neg$
Funksjoner:	$0 : Zero$
	$s : Nat \rightarrow NzNat$
	$p : Neg \rightarrow NzNeg$

der s og p står for hhv. "etterfølger-" og "forgjenger"-funksjonen.

La oss gå tilbake til $NatStack$-spesifikasjonen og anta at vi ønsker å evaluere uttrykket

$$top(pop(push(push(empty, 5), 8)))$$

som intuitivt gir mening selv om $pop(push(push(empty, 5), 8))$ egentlig er av typen $Stack$ og dermed ikke kan være et argument til *top*. I typeordnet algebra kan man "la tvilen komme uttrykket til gode" ved å innføre såkalte *retrakter*, som gir et uttrykk en lavere type hvis mulig. En retraktdeklarasjon er på formen

$$r_{s>s'} : s \rightarrow s'$$

der $s' < s$. Til enhver retrakt hører en *retraktlikning*

$$r_{s>s'}(x) \approx x \qquad \text{hvis } x \text{ er av type } s'$$

I vårt tilfelle ville vi hatt en retrakt $r_{Stack>NeStack} : Stack \rightarrow NeStack$ slik at termen

$$top(r_{Stack>NeStack}(pop(push(push(empty, 5), 8))))$$

er velformet. Dette beregnes til $top(r_{Stack>NeStack}(push(empty, 5)))$ og da $push(empty, 5)$ er av typen $NeStack$ kan vi bruke retraktlikningen som gir uttrykket $top(push(empty, 5))$.

Prøver vi optimistisk å beregne uttrykket

$$top(r_{Stack>NeStack}(pop(push(empty, 6))))$$

blir dette evaluert til $top(r_{Stack>NeStack}(empty))$, slik at vi aldri blir kvitt retrakten. Hvis tvilen ikke har noen mulighet til å komme utrykket til gode, kan vi avbryte med en informativ feilmelding.

Retrakter danner en sunn utvidelse til teorien når vi ikke tillater tomme typer, noe som gjør at retrakter og retraktlikninger automatisk kan legges til enhver spesifikasjon.

Jeg vil i dette kapitlet redegjøre for grunnbegrepene i typeordnet algebra; signaturer, tolkende algebraer og utledbarhet av gyldige likninger ut i fra en gitt mengde aksiomer. Kapitlet vil motivere hvorfor man alltid jobber med signaturer som er *regulære* (eller har tilsvarende egenskaper) under arbeid med termalgebraer. De to neste kapitlene omhandler hhv. typeordnet termomskrivning og typeordnet komplettering. Disse kapitlene motiverer hvorfor jeg bare beskjeftiger meg med regler som er *typeminskende*.

I motsetning til utypet algebra verserer det forskjellige definisjoner på typeordnede algebraer og typeordnet omskrivning. Mitt inntrykk er at basisartikkelen om typeordnede algebraer er skrevet av Goguen og Meseguer (i et utall versjoner av [GM88]), og det er denne de fleste, som Gnaedig et al., bygger på. Waldmann [Wal92] har i utgangspunktet mer liberale definisjoner, men må av og til forsterke kravene. Jeg vil derfor ta utgangspunkt i [GM88] i denne oppgaven.

3.1 Typeordnet signatur

En refleksiv partiell ordning $\leq$ over en mengde S er en binær relasjon over S som er refleksiv, transitiv og anti-symmetrisk (dvs. $x \leq y \wedge y \leq x \Longrightarrow x = y$).

Typeordnede signaturer bygger på en refleksiv partiell ordning $\leq$ over en mengde symboler kalt *typer*[2]. Relasjonen $<$ indusert av $\leq$ defineres $< \stackrel{d}{=} (\leq \setminus =)$ og relasjonen $\leq$ over n-tupler av S defineres

$$\langle s_1, \ldots, s_n \rangle \leq \langle s'_1, \ldots, s'_n \rangle \Longleftrightarrow \forall i \in 1..n | s_i \leq s'_i$$

[2] Jeg velger å oversette det engelske ordet "sort" med *type*.

Jeg skriver $\geq$ for den symmetriske relasjonen til $\leq$ og $s \bowtie t$ dersom hverken $s \leq t$ eller $t \leq s$. Jeg vil oftest benytte meg av symbolene $s, s_1, s_2, s', s'', \ldots$ for typer og symbolene $w, w', w'', w_1, \ldots$ for (tomme eller ikke-tomme) sekvenser av typer. Jeg skriver S^* for mengden av alle sekvenser av S-elementer.

Definisjon 3.1 (Typeordnet signatur) *En* typeordnet signatur *er et trippel* $\langle S, \leq, \Sigma \rangle$ *der* S *er en mengde symboler kalt typer,* $\leq$ *en refleksiv partiell ordning over* S *og* Σ *en mengde* $\{\Sigma_{w \to s} | w \in S^*, s \in S\}$ *av mengder. Et element i en mengde* $\Sigma_{w \to s}$ *kalles et funksjonssymbol. Når* $f \in \Sigma_{w \to s}$ *sier vi at* f *har* aritet w, rang $\langle w, s \rangle$ *og* (verdi-)type s, *og at* $f : w \to s$ *er en* deklarasjon *i* Σ.

Jeg skriver ofte Σ *for* $\langle S, \leq, \Sigma \rangle$ *når* S *og* $\leq$ *er gitt ut i fra sammenhengen.*

Et funksjonssymbol f kan som synes ha flere ariteter og verdityper, slik at $+$ f. eks. kunne være deklarert

$$+ : Int, Int \to Int$$
$$+ : Nat, Nat \to Nat$$

Uten den nederste deklarasjonen ville ikke et uttrykk $(a + b)!$ være velformet når a og b er av type Nat og $!$ er deklarert $! : Nat \to Nat$.

Eksempel 3.2 Jeg vil bruke følgende notasjon for å beskrive signaturer:

Typer: s, s_1, s_2, s_{12}
$s_1 \leq s$
$s_2 \leq s$
$s_{12} \leq s_1$
$s_{12} \leq s_2$

Funksjoner: $f : s_1 \to s_1$
$f : s_2 \to s_2$
$g : s \to s$
$g : s_1 \to s_2$
$a : s$
$b : s_1$
$c : s_2$
$d : s_{12}$

Mengden S av typer består av elementene s, s_1, s_2, s_{12} og relasjonen $\leq$ er den refleksive-transitive tillukningen av relasjonen $\leq$ beskrevet over. Mengden $\Sigma = \{\Sigma_{w \to s} | w \in S^*, s \in S\}$ der $\Sigma_{s_1 \to s_1} = \Sigma_{s_2 \to s_2} = \{f\}$, $\Sigma_{s \to s} = \Sigma_{s_1 \to s_2} = \{g\}$, $\Sigma_{\to s} = \{a\}$, $\Sigma_{\to s_1} = \{b\}$, $\Sigma_{\to s_2} = \{c\}$, $\Sigma_{\to s_{12}} = \{d\}$ og alle andre $\Sigma_{w \to s} = \emptyset$.

Jeg skriver ofte $f : s_1 \to s_1 \in \Sigma$ for $f \in \Sigma_{s_1 \to s_1}$. I dette tilfellet har f aritet s_1, rang $\langle s_1, s_1 \rangle$ og verditype s_1. Funksjonssymbolene i signaturen er a, b, c, d, f og g der de fire første også kalles konstanter.

Merk at definisjonen over ikke stiller noen spesielle konsistenskrav til en typeordnet signatur. Eksempelvis kunne $a : s_2 \to s_{12}$ vært en deklarasjon i signaturen over. Goguen og

Meseguer [GM88] krever i sin definisjon at typeordnede signaturer skal være *monotone*, dvs.

$$f \in (\Sigma_{w' \to s'} \cap \Sigma_{w \to s}) \wedge w' \leq w \implies s' \leq s$$

men dagens teoretikere forlanger av forskjellig grunner ofte *ikke* eksplisitt monotoni av signaturene. I praksis er forskjellen at ikke-monotone signaturer kan ha konstanter av flere typer mens monotonitetskravet gjør at en konstant ikke kan ha flere typer fordi

$$a \in \Sigma_{\epsilon \to s_1} \cap \Sigma_{\epsilon \to s_2} \wedge \epsilon \leq \epsilon \implies s_1 \leq s_2$$

og omvendt, slik at $s_1 = s_2$.

En typeordnet signatur $\langle S, \leq, \Sigma \rangle$ sies å være endelig dersom S er en endelig mengde og Σ bare inneholder et endelig antall deklarasjoner. En typeordnet signatur der relasjonen $\leq$ er lik $=$ kalles for en mangetypet signatur og en utypet signatur kan betraktes som en typeordnet signatur som bare inneholder én type.

3.2 Typeordnede algebraer

Signaturer gis mening av algebraer. Vi skiller mellom to former for typeordnede algebraer. I en *overlastet* algebra tolkes enhver *deklarasjon*, dvs. enhver linje på formen $f : w \to s$ i signaturen, av en funksjon, mens en *ikke-overlastet* algebra tolker ethvert *funksjonssymbol* som én funksjon.

Definisjon 3.2 (Overlastet typeordnet algebra) *La $\langle S, \leq, \Sigma \rangle$ være en typeordnet signatur. En* overlastet *typeordnet $\langle S, \leq, \Sigma \rangle$-algebra $\mathcal{A}$ består av en mengde $\{\mathcal{A}_s | s \in S\}$ av mengder og en funksjon $\mathcal{A}_f^{w \to s}$ for ethvert funksjonssymbol $f \in \Sigma_{w \to s}$ slik at*

1. $s' \leq s \implies \mathcal{A}_{s'} \subseteq \mathcal{A}_s$
2. $\mathcal{A}_f^{w' \to s'}$ *er lik* $\mathcal{A}_f^{w \to s}$ *på* $\mathcal{A}_{w'}$ *dersom* $w' \leq w$

når jeg skriver $\mathcal{A}_{s_1, \ldots, s_n}$ for $\mathcal{A}_{s_1} \times \cdots \times \mathcal{A}_{s_n}$.

Eksempel (forts. av eks. 3.2) La signaturen $\langle S, \leq, \Sigma \rangle$ være gitt som i eksempel 3.2. En overlastet Σ-algebra $\mathcal{A}$ kan da være definert ved domenene $\mathcal{A}_s = \mathcal{Z}$ (de hele tallene), $\mathcal{A}_{s_1} = \{\ldots, -2, -1, 0\}$, $\mathcal{A}_{s_2} = \{0, 1, 2, \ldots\}$ og $\mathcal{A}_{s_{12}} = \{0\}$ slik at krav 1 til overlastede algebraer er oppfylt. Funksjonssymbolet f kan tolkes

$$\mathcal{A}_f^{s_1 \to s_1} = \lambda x \in \mathcal{A}_{s_1} : x - 5 \text{ og}$$
$$\mathcal{A}_f^{s_2 \to s_2} = \lambda x \in \mathcal{A}_{s_2} : x + 10$$

og funksjonssymbolet g kan tolkes

$$\lambda x \in \mathcal{Z} : -x$$

som gjør at krav 2 er oppfylt. Konstantene a, b, c, d kan tolkes som hhv. 10, -2, 1 og 0. Man kan her legge merke til at termen $f(d)$ kan gis forskjellige "tolkninger", hhv. -5 og 10.

Definisjon 3.3 (Ikke-overlastet typeordnet algebra [Wal92]) *La $\langle S, \leq, \Sigma\rangle$ være en typeordnet signatur. En* ikke-overlastet *typeordnet $\langle S, \leq, \Sigma\rangle$-algebra $\mathcal{A}$ består av en mengde $\{\mathcal{A}_s | s \in S\}$ av mengder og en funksjon $\mathcal{A}_f : \mathcal{D}_f^{\mathcal{A}} \rightarrow \mathcal{C}_{\mathcal{A}}$ for ethvert funksjonssymbol $f \in \Sigma$ slik at*

1. $s' \leq s \Longrightarrow \mathcal{A}_{s'} \subseteq \mathcal{A}_s$
2. $\mathcal{D}_f^{\mathcal{A}} \subseteq (\mathcal{C}_{\mathcal{A}})^*$ *der* $\mathcal{C}_{\mathcal{A}} = \bigcup_{s \in S} \mathcal{A}_s$
3. *Hvis* $f \in \Sigma_{w \rightarrow s}$ *så er* $\mathcal{A}_w \subseteq \mathcal{D}_f^{\mathcal{A}}$ *og* $\mathcal{A}_f(\mathcal{A}_w) \subseteq \mathcal{A}_s$

Definisjonen kan ved første øyekast se litt tung ut, men forskjellen fra overlastet algebra ligger i at enhver $f \in \Sigma$ og *ikke* enhver $f \in \Sigma_{w \rightarrow s}$ tolkes som én funksjon. Denne funksjonen $\mathcal{A}_f$ er slik at dersom $f \in \Sigma_{w \rightarrow s}$ så er $\mathcal{A}_f$ definert for alle $\langle a_1, \ldots, a_n\rangle \in \mathcal{A}_w$ og funksjonsverdien $\mathcal{A}_f(a_1, \ldots, a_n)$ er naturlig nok et element i $\mathcal{A}_s$.

Eksempel (forts. av eks. 3.2) La Σ være signaturen i eksempel 3.2. Algebraen $\mathcal{A}$ der er ikke en ikke-overlastet Σ-algebra, fordi symbolet f ikke tolkes som én funksjon.

Signaturen kan imidlertid tolkes av en ikke-overlastet algebra $\mathcal{B}$ med samme domener som $\mathcal{A}$. Symbolene f og g kan tolkes som hhv. $\mathcal{B}_f = \lambda x \in \mathcal{Z} : x^3$ og $\mathcal{B}_g = \lambda x \in \mathcal{Z} : -x$ som tilfredsstiller kravene 2 og 3 til ikke-overlastede algebraer.

En ikke-overlastet algebra oppfyller alltid kravene til en overlastet algebra. En uheldig konsekvens av å se på ikke-overlastede algebraer er at mangetypet algebra ikke lenger blir et spesialtilfelle av typeordnet algebra (fordi en mangetypet algebra kan ha et element som tilhører flere typer). Det er mye lettere å jobbe med ikke-overlastede algebraer. Jeg vil i likhet med de fleste teoretikerne bare ta for meg overlastede algebraer. Den interesserte leser kan finne en grundig behandling av ikke-overlastede algebraer i [Wal92].

Homomorfier spiller en viktig rolle i teorier om algebraer og typeordnet homomorfi defineres på følgende måte i [GM88]:

Definisjon 3.4 (Typeordnet homomorfi) *La $\mathcal{A}$ og $\mathcal{B}$ være to (overlastede) Σ-algebraer. En $\langle S, \leq, \Sigma\rangle$-homomorfi $\phi : \mathcal{A} \rightarrow \mathcal{B}$ er en mengde $\phi = \{\phi_s : \mathcal{A}_s \rightarrow \mathcal{B}_s | s \in S\}$ av funksjoner slik at*

1. $\phi_{s'}$ *er lik* ϕ_s *på* $\mathcal{A}_{s'}$ *dersom* $s' \leq s$
2. $\phi_s(\mathcal{A}_f^{w \rightarrow s}(a_1, \ldots, a_n)) = \mathcal{B}_f^{w \rightarrow s}(\phi_{s_1}(a_1), \ldots, \phi_{s_n}(a_n))$ *for alle* $f \in \Sigma_{w \rightarrow s}$, *der* $w = s_1, \ldots, s_n$ *og* $\langle a_1, \ldots, a_n\rangle \in \mathcal{A}_w$

En homomorfi ϕ er henholdsvis injektiv og surjektiv dersom enhver komponent ϕ_s er det. En injektiv og surjektiv homomorfi kalles en isomorfi, men den "komponentvise" definisjonen av begrepet gir som vi skal se noen overraskende egenskaper til isomorfe algebraer.

En typeordnet algebra som er spesielt interessant for databehandlere er termalgebraen, som under visse forutsetninger er den meste generelle Σ-algebraen.

Definisjon 3.5 (Typeordnet grunntermalgebra) *Gitt en typeordnet signatur $\langle S, \leq, \Sigma\rangle$ er mengden $\mathcal{T}(\Sigma)_s$ av grunntermer med type s den minste mengden som tilfredsstiller*

1. *$a \in \mathcal{T}(\Sigma)_s$ hvis $a \in \Sigma_{\rightarrow s'}$ og $s' \leq s$*
2. *$f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma)_s$ dersom $f \in \Sigma_{s_1,\ldots,s_n \rightarrow s'}$ og $t_i \in \mathcal{T}(\Sigma)_{s_i}$ for alle $i \in 1 \ldots n$ og $s' \leq s$.*

Mengden av grunntermer $\mathcal{T}(\Sigma)$ er lik $\bigcup_{s \in S} \mathcal{T}(\Sigma)_s$, og et funksjonssymbol f tolkes på vanlig måte slik at $\mathcal{T}(\Sigma)_f^{w \rightarrow s}(t_1, \ldots, t_n)$ er lik termen $f(t_1, \ldots, t_n)$.

En typeordnet variabelmengde X er en mengde $\{X_s | s \in S\}$ av disjunkte mengder der jeg ofte skriver $x : s$ for $x \in X_s$. Når X og Σ er disjunkte defineres signaturen $\langle S, \leq, \Sigma \cup X\rangle$ ved $(\Sigma \cup X)_{\rightarrow s} = \Sigma_{\rightarrow s} \cup X_s$ og $(\Sigma \cup X)_{w \rightarrow s} = \Sigma_{w \rightarrow s}$ for $w \neq \epsilon$. Termalgebraen indusert av $\langle S, \leq, \Sigma \cup X\rangle$ skriver jeg $\mathcal{T}(\Sigma, X)$. Jeg vil anta at variabelmengdene inneholder variable av enhver type og vil ikke nevne disse spesielt når det ikke er nødvendig.

Et element i mengden $\mathcal{T}(\Sigma, X)$ vil jeg ofte kalle en *velformet* term, mens jeg av og til sier at et syntaktisk objekt er en ikke-velformet term dersom objektet ikke er med i mengden $\mathcal{T}(\Sigma, X)$.

Definisjon 3.6 (Variabeltilordning) *Gitt en typeordnet signatur $\langle S, \leq, \Sigma\rangle$ så er en variabeltilordning v fra en variabelmengde X til en Σ-algebra $\mathcal{A}$ en mengde funksjoner $\{v_s : X_s \rightarrow \mathcal{A}_s | s \in S\}$.*

3.3 Initialitet og regularitet

Den typeordnede termalgebraen $\mathcal{T}(\Sigma)$ er ikke nødvendigvis initiell i klassen av Σ-algebraer. I dette avsnittet defineres derfor *regularitet*, da $\mathcal{T}(\Sigma)$ er initiell i klassen av Σ-algebraer når signaturen Σ er regulær.

3.3.1 Initialitet

Initialalgebraer defineres på samme måte som i det utypede tilfellet, nemlig:

Definisjon 3.7 (Initialalgebra) *La Σ være en typeordnet signatur. En typeordnet Σ-algebra $\mathcal{A}$ er* initiell *i klassen av alle typeordnede Σ-algebraer hvis og bare hvis det fins én og bare én typeordnet homomorfi ϕ fra $\mathcal{A}$ til $\mathcal{B}$ for enhver typeordnet Σ-algebra $\mathcal{B}$.*

Det er ikke så interessant å arbeide med termalgebraen $\mathcal{T}(\Sigma)$ for dens egen skyld. Grunnen til at man (i utypet) algebra jobber med termalgebraen $T(F)$ er at egenskaper i $T(F)$ lett kan overføres til alle F-algebraer, fordi enhver term har én og bare én tolkning i en F-algebra A. Termalgebraen $T(F)$ er initiell i klassen av utypede F-algebraer og kan sies å representere alle F-algebraer.

Vi ønsker selvsagt at $\mathcal{T}(\Sigma)$ skal være initiell i klassen av alle typeordnede Σ-algebraer, men følgende eksempel viser at dette ikke er tilfelle:

Eksempel 3.3 La $\langle S, \leq, \Sigma \rangle$ være gitt ved

$$
\begin{array}{ll}
\textbf{Typer:} & s, s_1, s_2, s_{12} \\
& s_1 \leq s \\
& s_2 \leq s \\
& s_{12} \leq s_1 \\
& s_{12} \leq s_2 \\
\textbf{Funksjoner:} & f : s_1 \rightarrow s_1 \\
& f : s_2 \rightarrow s_2 \\
& a : s_{12}
\end{array}
$$

Man kan umiddelbart lure på hvor termen $f(a)$ "hører hjemme".

La $\mathcal{A}$ være en Σ-algebra der $\mathcal{A}_s = \{0, 1, 2\}$, $\mathcal{A}_{s_1} = \{0, 1\}$, $\mathcal{A}_{s_2} = \{1, 2\}$ og $\mathcal{A}_{s_{12}} = \{1\}$. Naturlig nok er da $\mathcal{A}_a^{s_{12}} = 1$. Vi lar så f tolkes av funksjonene $\mathcal{A}_f^{s_1 \rightarrow s_1} = \lambda x \in \{0, 1\} : 0$ og $\mathcal{A}_f^{s_2 \rightarrow s_2} = \lambda x \in \{1, 2\} : 2$.

Termen $f(a)$ kan nå tolkes som enten 0 eller 2 og en slik ambivalens er høyst uønsket. Mer formelt fins det ingen homomorfi ϕ fra $\mathcal{T}(\Sigma)$ til $\mathcal{A}$ fordi en slik homomorfi $\phi = \{\phi_s | s \in S\}$ måtte tilfredsstille $\phi_s = \phi_{s_1}$ på $\mathcal{T}(\Sigma)_{s_1}$ og $\phi_s = \phi_{s_2}$ på $\mathcal{T}(\Sigma)_{s_2}$. Tilsammen gir dette at ϕ må oppfylle $\phi_s(f(a)) = \phi_{s_1}(f(a)) = \mathcal{A}_f^{s_1 \rightarrow s_1}(\phi_{s_{12}}(a)) = 0$ og $\phi_s(f(a)) = \phi_{s_2}(f(a)) = \mathcal{A}_f^{s_2 \rightarrow s_2}(\phi_{s_{12}}(a)) = 2$, som er umulig da ϕ_s skal være en funksjon.

Tvetydigheten som blir illustrert av eksempel 3.3 er sannsynligvis også én av grunnene til at programmeringsspråket Simula ikke tillater multippelt prefikserte klasser. Anta nemlig at

$$
\begin{array}{r}
s \ \textbf{class} \ s_1; \ \ldots \ ; \\
s \ \textbf{class} \ s_2; \ \ldots \ ; \\
s_1, s_2 \ \textbf{class} \ s_{12}; \ \ldots \ ;
\end{array}
$$

var tre Simula-klassedeklarasjoner og anta at funksjonen f var deklarert inne i klassene s_1 og s_2 og at a var en peker til et s_{12}-objekt. Uttrykket $a.f$ ville da (i de aller fleste tilfeller) ikke vært veldefinert.

3.3.2 Regularitet

Det er liten vits i å jobbe med termalgebraer som er tvetydige på denne måten. Man kan ikke engang fastslå at identiteten $t \approx t$ er gyldig i alle Σ-algebraer, og da står man igjen med veldig lite hvis man skal resonnere om Σ-algebraer generelt ved hjelp av termalgebraen $\mathcal{T}(\Sigma, X)$. Man må derfor stille tilleggskrav til signaturen Σ for at $\mathcal{T}(\Sigma)$ skal bli Σ-initiell. Gougen og Meseguer [GM88] innførte kravet om regularitet som også Gnaedig, Kirchner og Kirchner bruker.

Definisjon 3.8 (Regulær signatur) *La $\langle S, \leq, \Sigma \rangle$ være en typeordnet signatur og anta at $f \in \Sigma_{w'' \rightarrow s''}$. Signaturen $\langle S, \leq, \Sigma \rangle$ er* regulær *hvis og bare hvis det for alle sekvenser av typer $w' \in S^*$ der $w' \leq w''$ fins en* éntydig minste *rang $\langle w, s \rangle \in S^* \times S$ slik at $w' \leq w$ og $f \in \Sigma_{w \rightarrow s}$.*

Eksempel (forts. eks. 3.3) Signaturen i eksempel 3.3 var ikke regulær, fordi gitt $s_{12} \leq s_1$ og gitt $f \in \Sigma_{s_1 \to s_1}$ så skulle det finnes en éntydig *minste* rang $\langle w, s' \rangle$ slik at $f \in \Sigma_{w \to s'}$ og $s_{12} \leq w$. Denne minste rangen fins ikke fordi både $\langle s_1, s_1 \rangle$ og $\langle s_2, s_2 \rangle$ er minimale f-ranger, samtidig som ingen av disse er éntydig minst.

For å få signaturen regulær, måtte signaturen inneholde deklarasjonen

$$f : s_{12} \to s_{12}$$

La Σ' være Σ utvidet med denne deklarasjonen. Algebraen $\mathcal{A}$ beskrevet tidligere er ikke lenger en Σ'-algebra, fordi $\mathcal{A}_f^{s_1 \to s_1}$ måtte være lik $\mathcal{A}_f^{s_{12} \to s_{12}}$ på $\mathcal{A}_{s_{12}}$ som i sin tur måtte være lik $\mathcal{A}_f^{s_2 \to s_2}$ på samme $\mathcal{A}_{s_{12}}$.

En algebra $\mathcal{A}'$ som tilfredsstiller kravene over har altså

$$\mathcal{A}'^{s_1 \to s_1}_f(\mathcal{A}'^{s_{12}}_a) = \mathcal{A}'^{s_{12} \to s_{12}}_f(\mathcal{A}'^{s_{12}}_a) = \mathcal{A}'^{s_2 \to s_2}_f(\mathcal{A}'^{s_{12}}_a) = \phi(f(a))$$

Termen $f(a)$ representerer altså et éntydig element i $\mathcal{A}'$ uansett "hvilken" f man velger. Det er lett å se at det fins én og bare én homomorfi ϕ fra $\mathcal{T}(\Sigma)$ til en slik $\mathcal{A}'$.

Følgende lemma gir bedre intuisjon om hva regularitet er:

Lemma 3.9 ([GM88]) *Anta at $\langle S, \leq, \Sigma \rangle$ er en typeordnet signatur der det ikke finnes noen uendelig kjede*

$$s_1 > s_2 > s_3 > \cdots$$

av typer. Da er signaturen regulær bare hvis[3] *det for alle funksjonssymboler f i Σ der $f \in \Sigma_{w_1 \to s_1} \cap \Sigma_{w_2 \to s_2}$ og det fins en w_{12} slik at $w_{12} \leq w_1, w_2$ så fins det en w og s slik at $w_{12} \leq w \leq w_1, w_2$ og $f \in \Sigma_{w \to s}$.*

Regularitet fjerner altså tvetydigheten i tolkningen av termene, og medfører at termalgebraen kan brukes til å fastslå egenskaper for alle Σ-algebraer fordi

Teorem 3.10 ([GM88]) *Hvis Σ er en regulær signatur, så er termalgebraen $\mathcal{T}(\Sigma)$ initiell i klassen av alle typeordnede Σ-algebraer.*

Definisjon 3.11 (Minste type (LS)) *Gitt en typeordnet signatur $\langle S, \leq, \Sigma \rangle$ har en term $t \in \mathcal{T}(\Sigma, X)$ en éntydig minste type, kalt $LS(t)$ der $LS(t) \in S$, hvis og bare hvis*

$$\forall s \in S \mid t \in \mathcal{T}(\Sigma, X)_s \Longrightarrow LS(t) \leq s$$

Når Σ er regulær har enhver term $t \in \mathcal{T}(\Sigma, X)$ har en minste type $LS(t)$ (se [GM88], [GKK90]). Denne minste typen er effektivt beregnbar ved at $LS(x : s) = s$ og $LS(f(t_1, \ldots, t_n)) = s$ der s er den minste typen slik at det fins en deklarasjon $f : s_1, \ldots, s_n \to s$ i Σ slik at $t_i \in \mathcal{T}(\Sigma, X)_{s_i}$ for alle i.

Dersom Σ er regulær er også $\mathcal{T}(\Sigma, X)$ en fri typeordnet Σ-algebra over X, det vil si at enhver tilordning $v : X \to \mathcal{A}$ til en algebra $\mathcal{A}$ kan utvides éntydig til en homomorfi $v^* : \mathcal{T}(\Sigma, X) \to \mathcal{A}$.

[3] For å kunne påstå "hvis og bare hvis" her som [GM88] gjør, måtte vi antatt at signaturen var monoton.

Som jeg allerede har nevnt er det ofte lite ønskelig å arbeide med termalgebraer som ikke er initielle. Alle som skriver om typeordnet spesifikasjon eller termomskrivning antar helt i begynnelsen at signaturen Σ innehar egenskaper som gjør at $\mathcal{T}(\Sigma)$ er initiell i klassen av Σ-algebraer.

Vanligvis antas regularitet, men Waldmann viser i [Wal92] at et svakere krav til signaturer, nemlig den at enhver grunnterm bare har én tolkning i enhver Σ-algebra $\mathcal{A}$, fører til at $\mathcal{T}(\Sigma)$ er en Σ-initialalgebra. Denne tilnærmingsmåten gjør at konstanter kan ha flere typer og at mange egenskaper blir meget vanskelige å beregne. I kompletteringssammenheng må imidlertid også Waldmann forlange regulære signaturer! Jeg velger å ikke arbeide videre med "Waldmann-signaturer".

3.3.3 Preregularitet

Enkelte artikler om termomskrivning forlanger bare *preregularitet*, og ikke regularitet, av signaturer.

Definisjon 3.12 (Preregulær signatur) *En signatur Σ er* preregulær *hvis og bare hvis enhver term t i $\mathcal{T}(\Sigma, X)$ har en éntydig minste type når X inneholder minst én variabel av enhver type.*

Preregularitet sikrer imidlertid ikke initialitet av $\mathcal{T}(\Sigma)$, som følgende eksempel viser:

Eksempel (forts. av eks. 3.3) Når man bytter ut deklarasjonene

$$f : s_1 \rightarrow s_1 \text{ og } f : s_2 \rightarrow s_2$$

med deklarasjonene

$$f : s_1 \rightarrow s \text{ og } f : s_2 \rightarrow s$$

i signaturen i eksempel 3.3 har alle termer en éntydig minste type. Funksjonen(e) f kan imidlertid tolkes som tidligere, slik at termen $f(a)$ fremdeles kan tolkes enten 0 eller 2.

Én av grunnene til at enkelte forfattere (i f. eks. [CH93]) nøyer seg med å anta preregularitet er at dette i mange tilfeller er en tilstrekkelig egenskap for å vise ønskede resultater eller fremgangsmåter. Siden enhver regulær signatur også er preregulær betyr det at viste resultater holder for regulære signaturer, samtidig som man holder døren på gløtt for eventuelle andre, mindre strenge krav som gjør $\mathcal{T}(\Sigma)$ initiell.

Også for meg vil preregularitet være (både nødvendig og) tilstrekkelig i mitt videre arbeid.

3.3.4 Monotoni

Monotonikravet til signaturer i [GM88] er som nevnt utelatt i andre definisjoner av typeordnede signaturer av typeordnede signaturer. Grunnen er enten (som hos Waldmann i [Wal92]) at man virkelig vil omhandle signaturer som ikke oppfyller dette kravet, eller (som vel er tilfelle hos Gnaedig, Kirchner & Kirchner ([Gn92a] og [GKK90]) og Chen &

Hsiang [CH93]) at monotoni ligger implisitt fordi det følger fra preregularitet og en grei "presisering" av signaturen.

Den preregulære signaturen i følgende eksempel kan gjøres monoton med en enkel presisering av deklarasjonene:

Eksempel 3.4 I signaturen

Typer:	Int, Nat
	$Nat \leq Int$
Funksjoner:	$f : Int \rightarrow Nat$
	$f : Nat \rightarrow Int$
	$a : Int$
	$b : Nat$

har enhver term en éntydig minste type slik at signaturen er preregulær, men langt fra monoton. Deklarasjonen $f : Nat \rightarrow Int$ virker litt underlig siden siden deklarasjonen $f : Int \rightarrow Nat$ og krav 2 til typeordnede algebraer sikrer $\mathcal{A}_f^{Nat \rightarrow Int}(\mathcal{A}_{Nat}) \subseteq \mathcal{A}_{Nat}$ for enhver tolkende algebra $\mathcal{A}$. Deklarasjonen $f : Nat \rightarrow Int$ kunne like gjerne vært fjernet eller erstattet med $f : Nat \rightarrow Nat$ i lys av dette.

En deklarasjon $f : w \rightarrow s$ kan derfor sies å være *for slapp* dersom det fins en deklarasjon $f : w' \rightarrow s'$ der $w \leq w'$ og $s' < s$. En slik for slapp deklarasjon kan uten tvil erstattes av en deklarasjon $f : w \rightarrow s'$ uten å forandre sentrale egenskaper ved signaturen. En preregulær signatur som er *strammet inn* på denne måten er monoton.

Jeg vil senere ha nytte av følgende proposisjon:

Proposition 3.13 *Anta at Σ er en typeordnet signatur. Hvis Σ er preregulær og ikke inneholder for slappe deklarasjoner så er Σ monoton.*

Bevis: Dersom Σ ikke inneholder for slappe deklarasjoner og ikke er monoton, må det finnes to deklarasjoner $f : w' \rightarrow s'$ og $f : w \rightarrow s$ slik at $w' \leq w$ og $s' \bowtie s$. Preregularitet sier imidlertid at termen $f(\overline{x} : w')$ må ha en minste type $s'' < s', s$. Altså må det finnes en deklarasjon $f : w'' \rightarrow s''$ der $w' \leq w''$. Siden $s'' < s'$ og $w' \leq w''$ er imidlertid deklarasjonen $f : w'' \rightarrow s''$ for slapp og eksisterer dermed ikke. □

3.4 Likninger

En typeordnet spesifikasjon $\langle \Sigma, E \rangle$ består av en typeordnet signatur Σ og en mengde E av likninger $(\forall X)\ t \approx t'$ over termer i $\mathcal{T}(\Sigma, X)$.

3.4.1 Typene til termene i en likning

Termene t og t' i en likning $t \approx t'$ trenger ikke ha samme minste type, da dette ville diskvalifisere en likning(instans) som $pop(push(empty, 5)) \approx empty$. Man må derfor tillate at $LS(t) \leq LS(t')$ eller $LS(t') \leq LS(t)$ og følgende eksempel viser at dette fører til typene til to termer i en likning ikke engang trenger å være sammenlignbare.

Eksempel 3.5 ([GM88]) La

Typer:	s, s_1, s_2, s'
	$s_1 \leq s$
	$s_2 \leq s$
	$s_2 \leq s'$
Funksjoner:	$a : s_1$
	$b : s$
	$c : s_2$
	$d : s'$
Likninger:	$a \approx b$
	$b \approx c$
	$c \approx d$

være en typeordnet spesifikasjon $\langle \Sigma, E \rangle$. Enhver Σ-algebra $\mathcal{A}$ som tilfredsstiller $\phi(a) = \phi(b) = \phi(c) = \phi(d)$ tilfredsstiller $\phi(a) = \phi(d)$ for den éntydige (siden Σ er regulær) homomorfien $\phi : \mathcal{T}(\Sigma) \rightarrow \mathcal{A}$. Derfor burde (må![4]) $a \approx d$ kunne være en likning i spesifikasjonen, selv om typene til a og d ikke engang er sammenlignbare.

Eksemplet antyder at det er tilstrekkelig at termene i en likning har typer i samme *sammenhengende komponent* i $(S, \leq)$.

Definisjon 3.14 (Sammenhengende komponent) *Dersom S er en mengde og $\leq$ er en refleksiv partiell ordning over S, kalles ekvivalensklassene til den transitive tillukningen til* $(\leq \cup \geq)$ sammenhengende komponenter *til* $(S, \leq)$.

Når $(S, \leq)$ tegnes opp som en urettet graf, er to elementer s_1 og s_2 i samme sammenhengende komponent dersom det fins en sti fra s_1 til s_2 i grafen. I signaturen $\langle S, \leq, \Sigma \rangle$ i forrige eksempel er alle typene s, s', s_1, s_2 i samme sammenhengende komponent til $(S, \leq)$.

Definisjon 3.15 (Typeordnet likning) *Anta at Σ er en regulær typeordnet signatur. En Σ-likning er et trippel $\langle X, t, t' \rangle$, som skrives $(\forall X)\, t \approx t'$, der X er en typeordnet variabelmengde, t, t' er termer i $\mathcal{T}(\Sigma, X)$ og $LS(t)$ og $LS(t')$ ligger i samme sammenhengende komponent til $(S, \leq)$.*

3.4.2 Gyldighet

Vi er da klar for å definere gyldighet av likninger i typeordnede algebraer.

Definisjon 3.16 (Gyldighet) *En typeordnet Σ-algebra $\mathcal{A}$* tilfredsstiller *en likning $(\forall X)\, t \approx t'$ hvis og bare hvis $\phi^*(t) = \phi^*(t')$ for alle (éntydige) utvidelser av en tilordning $\phi : X \rightarrow \mathcal{A}$ til en homomorfi $\phi^* : \mathcal{T}(\Sigma, X) \rightarrow \mathcal{A}$. Algebraen $\mathcal{A}$ tilfredsstiller en mengde likninger E, og kalles en $\langle \Sigma, E \rangle$-algebra, hvis den tilfredsstiller enhver likning i E.*

Hvis en likning er tilfredsstilt i en algebra $\mathcal{A}$, sier vi også at likningen er gyldig i $\mathcal{A}$. *En likning som er tilfredsstilt av alle $\langle \Sigma, E \rangle$-algebraer kaller jeg bare (E-)*gyldig *eller en* logisk konsekvens *av E.*

[4] for å kunne oppnå kompletthet

Det følger av definisjonen at enhver likning på formen $(\forall x : s)\ t \approx t'$ er tilfredsstilt av en algebra $\mathcal{A}$ dersom mengden $\mathcal{A}_s$ er tom, slik at man må være oppmerksom på variabelmengdene på en helt annen måte i den utypede verdenen. Jeg vil komme tilbake til dette.

Den "komponentvise" definisjonen av isomorfi gjør at gyldighet dessverre ikke er lukket under isomorfi av algebraer, slik at en likning som er gyldig i en algebra $\mathcal{A}$ ikke nødvendigvis er gyldig i den isomorfe algebraen $\mathcal{B}$. Eksempelet nedenfor illustrerer dette:

Eksempel 3.6 ([GM88]) La den typeordnede spesifikasjonen $\langle \Sigma, E \rangle$ være

Typer:	s_1, s_2, s_{12}
	$s_{12} \leq s_1$
	$s_{12} \leq s_2$
Funksjoner:	$a : s_1$
	$b : s_{12}$
	$c : s_2$
Likning:	$a \approx c$

og la $\mathcal{A}$ være en Σ-algebra slik at både a og c tolkes som elementet D og b tolkes som B. Elementene i $\mathcal{A}$ er da gitt $\mathcal{A}_{s_{12}} = \{B\}$, $\mathcal{A}_{s_1} = \mathcal{A}_{s_2} = \{B, D\}$, mens elementene i termalgebraen er $\mathcal{T}(\Sigma)_{s_{12}} = \{b\}$, $\mathcal{T}(\Sigma)_{s_1} = \{a, b\}$ og $\mathcal{T}(\Sigma)_{s_2} = \{b, c\}$. Den éntydige homomorfien $\phi : \mathcal{T}(\Sigma) \to \mathcal{A}$ må da være $\{a \mapsto D,\ b \mapsto B,\ c \mapsto D\}$, som kanskje ikke ser ut som en isomorfi. Komponentene

$$\begin{aligned} \phi_{s_1} &: \mathcal{T}(\Sigma)_{s_1} \to \mathcal{A} = \{a \mapsto D, b \mapsto B\} \\ \phi_{s_2} &: \mathcal{T}(\Sigma)_{s_2} \to \mathcal{A} = \{c \mapsto D, b \mapsto B\} \text{ og} \\ \phi_{s_{12}} &: \mathcal{T}(\Sigma)_{s_{12}} \to \mathcal{A} = \{b \mapsto B\} \end{aligned}$$

er imidlertid alle isomorfier, slik at ϕ også er en isomorfi per definisjon. Termalgebraen $\mathcal{T}(\Sigma)$ tilfredsstiller ikke likningen $a \approx c$ fordi $a \neq c$, mens den isomorfe algebraen $\mathcal{A}$ tilfredstiller likningen fordi $\phi(a) = \phi(c) = D$.

For å rette opp dette kunne man forandre definisjonen av typeordnet isomorfi, men å forlange koherente signaturer gjør samme nytten.

Definisjon 3.17 (Koherent signatur) *En mengde S ordnet av en refleksiv partiell ordning $\leq$ er* filtrert *dersom $\forall s, s' \in S | \exists s'' \in S | s, s' \leq s''$. Mengden S er* lokalt filtrert *dersom enhver sammenhengende komponent til $(S, \leq)$ er filtrert.*

En typeordnet signatur $\langle S, \leq, \Sigma \rangle$ er koherent *dersom $(S, \leq)$ er lokalt filtrert og Σ er regulær.*

Dersom S er endelig er $(S, \leq)$ lokalt filtrert hvis og bare hvis enhver sammenhengende komponent har et største element s. På denne måten kan vi si at "komponenten" ϕ_s står for "hele" isomorfien ϕ (i den sammenhengende komponenten) når s er det maksimale elementet.

For koherente signaturer er gyldighet nå lukket under isomorfi av algebraer:

Lemma 3.18 ([GM88]) *Gitt en koherent signatur Σ og to isomorfe Σ-algebraer $\mathcal{A}$ og $\mathcal{B}$ så er likningen $(\forall X)\ t \approx t'$ gyldig i $\mathcal{A}$ hvis og bare hvis likningen er gyldig i $\mathcal{B}$.*

Eksempel (forts. av eks. 3.6) Signaturen i eksempel 3.6 er ikke koherent, fordi det ikke finnes en s slik at $s_1, s_2 \leq s$. Legger vi til en slik maksimal type s til Σ, så er $\mathcal{T}(\Sigma)_s = \{a, b, c\}$ og $\mathcal{A}_s = \{B, D\}$ og ϕ kan da ikke lenger være en isomorfi fordi komponenten ϕ_s umulig kan være det.

Det er en enkel sak å gjøre en regulær signatur koherent ved å legge et "toppelement" til hver sammenhengende komponenet.

3.4.3 Utledning av likninger

Jeg vil i dette avsnittet anta at Σ er en koherent typeordnet signatur.

En variabelsubstitusjon σ er en variabeltilordning $\sigma : Y \rightarrow \mathcal{T}(\Sigma, X)$ og jeg skriver også σ (med postfiks notasjon) for den homomorfe utvidelsen $\sigma^* : \mathcal{T}(\Sigma, Y) \rightarrow \mathcal{T}(\Sigma, X)$ av σ.

En likning $(\forall X)\ t \approx t'$ sies å være *utledbar* fra en mengde likninger E, som jeg skriver $(\forall X)\ t \approx_E t'$ dersom $(\forall X)\ t \approx_E t'$ kan fås ved å bruke deduksjonsregelsettet i figur 3.1.

Refleksivitet: $$(\forall X)\ t \approx_E t$$

Symmetri: $$\frac{(\forall X)\ t \approx_E t'}{(\forall X)\ t' \approx_E t}$$

Transitivitet: $$\frac{(\forall X)\ t \approx_E t' \quad (\forall X)\ t' \approx_E t''}{(\forall X)\ t \approx_E t''}$$

Kongruens: $$\frac{\forall x \in X \mid (\forall Y)\ x\sigma \approx_E x\sigma'}{(\forall Y)\ t\sigma \approx_E t\sigma'}$$

Substitutivitet: $$(\forall Y)\ t\sigma \approx_E t'\sigma \qquad \text{dersom } (\forall X)\ t \approx t' \in E$$

for alle termer t, t', t'' i $\mathcal{T}(\Sigma, X)$ og alle substitusjoner $\sigma, \sigma' : X \rightarrow \mathcal{T}(\Sigma, Y)$.

Figur 3.1: *Deduksjonsregler for utledning av likninger gitt en mengde likninger E og en koherent signatur Σ.*

Kongruensregelen kan ved første øyekast se litt merkelig ut, men man kan legge merke til at

$$\frac{(\forall X)\ t_1 \approx_E t'_1\ ,\ \ldots\ ,\ (\forall X)\ t_n \approx_E t'_n \qquad \forall i \in 1..n | t_i, t'_i \in \mathcal{T}(\Sigma, X)_{s_i} \qquad f \in \Sigma_{s_1,\ldots,s_n \rightarrow s}}{(\forall X)\ f(t_1, \ldots, t_n) \approx_E f(t'_1, \ldots, t'_n)}$$

er en instans av denne med $t = f(x_1, \ldots, x_n)$, substitusjonene $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ og $\sigma' = \{x_1 \mapsto t'_1, \ldots, x_n \mapsto t'_n\}$. En likning i E er utledbar med den siste regelen og den tomme substitusjonen.

Vi skal i kapittel 5 filosofere mer over variabelmengdene, men som en kort begrunnelse om viktigheten av variabelmengdene kan vi anta at Tom er en tom type og T og F er

konstanter av typen *Bool*. Likningene $(\forall x : Tom)\ f(x) \approx_E F$ og $(\forall x : Tom)\ f(x) \approx_E T$ er trivielt gyldige i alle algebraer $\mathcal{A}$ med $\mathcal{A}_{Tom} = \emptyset$. Ved transitivitet kan vi *ikke* slutte den ugyldige likningen $T \approx_E F$, men den gyldige likningen $(\forall x : Tom)\ T \approx_E F$.

Med deduksjonsreglene i figur 3.1 kan man utlede alle likninger som er logiske konsekvenser av E:

Teorem 3.19 ([GM88]) *Dersom Σ er en koherent typeordnet signatur, E en mengde Σ-likninger og t, t' termer i $\mathcal{T}(\Sigma, X)$ så er*

$$(\forall X)\ t \approx_E t' \text{ utledbar fra } E \text{ ved deduksjonsreglene i figur 3.1}$$

hvis og bare hvis

$$(\forall X)\ t \approx t' \text{ er gyldig i enhver } \langle \Sigma, E \rangle\text{-algebra.}$$

Man kan legge til retraktlikninger på formen

$$(\forall x : s')\ r_{s>s'}(x) \approx x$$

til enhver typeordnet spesifikasjon hvis vi ikke tillater tomme typer (der en type s sies å være tom dersom en Σ-algebra $\mathcal{A}$ kan ha domenet $\mathcal{A}_s = \emptyset$). Retrakter kan sammen med tomme typer gjøre at man kan utlede ikke-gyldige likninger.

3.5 Oppsummering

For at det skal gi mening å jobbe med termalgebraen $\mathcal{T}(\Sigma, X)$ vil jeg anta at alle signaturene er regulære, men er åpen for enklere krav til initialitet av $\mathcal{T}(\Sigma)$ ved at min senere fremgangsmåte bare forlanger preregularitet.

For en grundigere innføring i typeordnede algebraer og spesifikasjoner anbefales [GM88] og [Wal92].

Kapittel 4

Typeordnet omskrivning

Typeordnet omskrivning er et felt som omfattes av stadig større interesse. Det fins som nevnt i ingressen t. o. m. et programmeringsspråk, OBJ3 [GW88], der et OBJ3-program ikke er noe annet enn et typeordnet omskrivningssystem. Typeordnet omskrivning kan, som utypet omskrivning, brukes til mye annet enn automatisk bevisføring og i forbindelse med algebraiske spesifikasjoner. Eksempel 6.1 i kapittel 6 viser et typisk eksempel på et typeordnet omskrivningssystem. Derfor vil jeg nå vise et eksempel på et problem som tilsynelatende ikke har så mye med typeordnet algebra å gjøre, men der problemet kan beskrives av et typeordnet omskrivningssystem, samtidig som det *ikke* kan beskrives på en enkel måte av et utypet system.

Eksempel 4.1 I *kaffebønnespillet* (beskrevet i [Kir92]) er det gitt en sekvens av hvite og svarte kaffebønner. To svarte bønner ved siden av hverandre kan erstattes med én hvit bønne og en hvit bønne ved siden av en svart kan fjernes i spillet, som går ut på å ende opp med færrest mulig bønner til slutt. I [Kir92] formuleres spillet som et strengomskrivningssystem[1] med regelmengden $\{ss \longrightarrow h,\ sh \longrightarrow s,\ hs \longrightarrow s\}$, der s og h representerer hhv. en svart og en hvit kaffebønne. Etter å spilt spillet noen ganger synes vi det er litt for lett, og ønsker at "regelen" $sh \longrightarrow s$ bare kan brukes når det *ikke* er noen svarte bønner til høyre for den hvite (selv om "regelen" $hs \longrightarrow s$ naturligvis må beholdes for at spillet skal være interessant). Det er meget vanskelig å representere denne nye varianten av spillet som et utypet omskrivningssystem, men spillet kan formuleres som et typeordnet termomskrivningssystem på følgende enkle måte:

Typer:	$Spill, BareHvite$
	$BareHvite \leq Spill$
Funksjoner:	$s : Spill \rightarrow Spill$
	$h : BareHvite \rightarrow BareHvite$
	$h : Spill \rightarrow Spill$
	$\Box : BareHvite$
Regler:	$(\forall x{:}Spill)\ s(s(x)) \longrightarrow h(x)$
	$(\forall x{:}BareHvite)\ s(h(x)) \longrightarrow s(x)$
	$(\forall x{:}Spill)\ h(s(x)) \longrightarrow s(x)$

[1] Se [Kir92] for en beskrivelse av strengomskrivningssystemer.

der $\Box$ er et "sluttmerke" slik at strengen $hsshs$ representeres av termen $h(s(s(h(s(\Box)))))$.

Eksemplet illustrerer også mer eksplisitt en annen av grunnene til den store interessen for typeordnet omskrivning, nemlig at den "bygger bro" mellom utypet og betinget omskrivning, siden vi i eksemplet har klart å formulere den betingede regelen

if alle bønnene til høyre er hvite **then** $sh \longrightarrow s$

i et ubetinget typeordnet system.

Hovedforskjellen mellom utypet og typeordnet omskrivning er at den sistnevnte relasjonen ikke er lukket under "kontekst-applikasjon i utypet forstand" som følgende system illustrerer:

Typer:	Int, Nat
	$Nat \leq Int$
Funksjoner:	$f : Nat \rightarrow Int$
	$a : Int$
	$b : Nat$
Regel:	$b \longrightarrow a$

Termen $f(b)$ kan her ikke skrives om til $f(a)$ fordi denne siste ikke er velformet.

Dette kapitlet vil omhandle typeordnet omskrivning generelt, mens neste kapittel vil ta for seg typeordnet komplettering og egenskaper som naturlig hører med under komplettering. Jeg vil peke på forskjeller mellom utypet og typeordnet omskrivning og komplettering uten å gå inn på spissfindigheter vi også kjenner fra utypet omskrivning.

Siden jeg omtaler *term*omskrivning antar jeg at alle signaturene er regulære; dette fører til at termalgebraen $\mathcal{T}(\Sigma, X)$ "representerer" alle Σ-algebraer.

4.1 Den typeordnede omskrivningsrelasjonen

En likning $(\forall X)\ l \approx r$ kalles en *omskrivningsregel* dersom den brukes operasjonelt i en bestemt retning.

Definisjon 4.1 (Omskrivningsregel ([GKK90])) *En* typeordnet omskrivningsregel *er en typeordnet likning* $(\forall X)\, l \approx r$ *som tilfredsstiller* $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ *og skrives* $(\forall X)\, l \longrightarrow r$.

I utypet omskrivning forlanger man ikke lenger $\mathcal{V}(r) \subseteq \mathcal{V}(l)$ og Waldmann ([Wal92]) sier også *"we do not restrict to rules where* $\mathcal{V}(r) \subseteq \mathcal{V}(l)$*, following [DJ90]"* . [DJ90] omhandler utypet omskrivning, og jeg vil i avsnitt 8.2 vise at det i dette henseende er en fundamental forskjell mellom utypet og typeordnet omskrivning.

Definisjon 4.2 (Typeordnet omskrivningssystem) *Et* typeordnet omskrivningssystem *er et par* $\langle \Sigma, \mathcal{R} \rangle$ *der* Σ *er en typeordnet signatur og* $\mathcal{R}$ *er en (endelig eller uendelig) mengde* $\bigcup_{i=1}^{n} \{l_i \longrightarrow r_i\}$ *av omskrivningsregler der* $\forall i \in 1..n \mid l_i, r_i \in \mathcal{T}(\Sigma, X)$ *for en typeordnet variabelmengde* X.

Jeg skriver ofte $\mathcal{R}$ *for* $\langle \Sigma, \mathcal{R} \rangle$ *når signaturen* Σ *fremgår av signaturen eller er uten interesse i sammenhengen.*

Vi har allerede sett at "kontekst-applikasjon" kan føre til at en velformet term "omskrives" til en ikke-velformet term. Omskrivningsrelasjonen må opplagt omskrive en velformet til en ny velformet term, men er dette kravet tilstrekkelig? La oss vurdere

Eksempel 4.2 La den koherente (dvs. regulære og lokalt filtrerte) signaturen Σ og regelmengden $\mathcal{R}$ være

Typer:	s, s_1, s_2
	$s_1 \leq s$
	$s_2 \leq s$
Funksjoner:	$f : s_1 \rightarrow s$
	$f : s_2 \rightarrow s$
	$a : s_1$
	$b : s_2$
Regel:	$a \longrightarrow b$

Spørsmålet man kan vurdere er hvorvidt $f(a) \xrightarrow[\mathcal{R}]{} f(b)$ skal være en omskrivning i $\mathcal{R}$ samtidig som vi registrerer at både $f(a)$ og $f(b)$ er velformede termer.

For å besvare spørsmålet må man se på hva et omskrivningssystem egentlig er ment å være. (Utypet) omskrivning bygger mye på "kontekst-applikasjonsbegrepet". Representerer $a \longrightarrow b$ forholdet $a \simeq b$ må også $t[a]_p \simeq t[b]_p$ gjelde dersom $t[a]_p \longrightarrow t[b]_p$.

Hvis $a \longrightarrow b$ fordi $\phi(a) = \phi(b)$ for alle homomorfier fra den aktuelle termelagebraen inn i enhver modell av systemet, så er også $\phi(t[a]_p) = \phi(t[b]_p)$ i alle tilsvarende utypede systemer. Omskrivningssystemer handler selvfølgelig ikke bare om likhetsrelasjoner, men uansett må det generelt gjelde at dersom $a \xrightarrow[\mathcal{R}]{} b$, så må "samme f" brukes på både $f(\ldots, a, \ldots)$ og $f(\ldots, b, \ldots)$.

Eksempel (forts. av eks. 4.2) La $\mathcal{A}$ være en tolkning av signaturen Σ i eksempel 4.2 gitt ved $\mathcal{A}_s = \mathcal{Z}$, $\mathcal{A}_{s_1} = \{\ldots, -2, -1, 0\}$ og $\mathcal{A}_{s_2} = \{0, 1, 2, \ldots\}$. Både a og b tolkes i $\mathcal{A}$ som elementet 0. Funksjonssymbolet f tolkes $\mathcal{A}_f^{s_1 \rightarrow s} = \lambda x \in \mathcal{A}_{s_1} : -12$ og $\mathcal{A}_f^{s_2 \rightarrow s} = \lambda x \in \mathcal{A}_{s_2} : +55$. Algebraen $\mathcal{A}$ er en typeordnet Σ-algebra.

Jeg vil i dette tilfellet at $s \xrightarrow[\mathcal{R}]{} t$ skal representere $\phi(s) = \phi(t)$ for den éntydige homomorfien $\phi : \mathcal{T}(\Sigma) \rightarrow \mathcal{A}$. Algebraen $\mathcal{A}$ tilfredsstiller dette ønsket fordi $\phi(a) = \phi(b) = 0$. Dersom $f(a) \xrightarrow[\mathcal{R}]{} f(b)$ så skulle også $\phi(f(a)) = \phi(f(b))$, men dette holder ikke fordi $\phi(f(a)) = -12$ og $\phi(f(b)) = 55$. Dermed bør vi ikke ha $f(a) \xrightarrow[\mathcal{R}]{} f(b)$ selv om $a \xrightarrow[\mathcal{R}]{} b$ og både $f(a)$ og $f(b)$ er velformede termer.

Siden $s_1 \bowtie s_2$ er det faktisk *ingen* sammenheng mellom $\mathcal{A}_f^{s_1 \rightarrow s}$ og $\mathcal{A}_f^{s_2 \rightarrow s}$. Deklarasjonen $f : s_2 \rightarrow s$ kunne like gjerne vært skrevet $g : s_2 \rightarrow s$. Utifra dette hadde det også vært unaturlig å ha en omskrivning $f(a) \xrightarrow[\mathcal{R}]{} f(b)$.

Problemet oppstår fordi et funksjonssymbol kan tolkes på forskjellig måte avhengig av typene til argumentene, selv om argumentene har samme verdi, i dette tilfellet 0. Det ville gått bra dersom $\mathcal{A}_f^{s_1 \rightarrow s}$ og $\mathcal{A}_f^{s_2 \rightarrow s}$ hadde gitt samme resultat for samme argument.

Eksempel (forts. av eks. 4.2) For å sikre at $\mathcal{A}_f^{s_1 \to s} = \mathcal{A}_f^{s_2 \to s}$ på $\mathcal{A}_{s_1} \cap \mathcal{A}_{s_2}$ må signaturen Σ utvides med en deklarasjon $f : s \to s$.

Tolkningen $\mathcal{A}_f^{s \to s}$ må da tilfredsstille $\mathcal{A}_f^{s \to s} = \mathcal{A}_f^{s_1 \to s}$ på $\mathcal{A}_{s_1}$ og $\mathcal{A}_f^{s \to s} = \mathcal{A}_f^{s_2 \to s}$ på $\mathcal{A}_{s_2}$ som gir ønskede $\mathcal{A}_f^{s_1 \to s} = \mathcal{A}_f^{s \to s} = \mathcal{A}_f^{s_2 \to s}$ på $\mathcal{A}_{s_1} \cap \mathcal{A}_{s_2}$.

Rent konkret ville det i vårt tilfelle bety at dersom $\phi(a) = \phi(b) = 0$ ville også $\mathcal{A}^{s_1 \to s}(0) = \mathcal{A}^{s_2 \to s}(0)$ slik at man kunne si $f(a) \underset{\mathcal{R}}{\longrightarrow} f(b)$ når man med $\underset{\mathcal{R}}{\longrightarrow}$ vil mene likhet i Σ-algebraer.

For at "samme f" blir brukt på $f(\ldots, a, \ldots)$ og $f(\ldots, b, \ldots)$ må det altså finnes en type s slik at $f(\ldots, x{:}s, \ldots)$ er veldefinert og a og b er med i $\mathcal{T}(\Sigma, X)_s$. Dette blir sikret av krav 2 til den typeordnede omskrivningsrelasjonen som defineres på følgende måte:

Definisjon 4.3 (Typeordnet omskrivningsrelasjon) *En term $t \in \mathcal{T}(\Sigma, X)$ omskrives til en term t' med en omskrivningsregel $(\forall Y)\ l \longrightarrow r$ i et omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$ i en posisjon p, som skrives $t \underset{\mathcal{R}}{\longrightarrow}_X t'$, når*

1. *det fins en substitusjon $\sigma : Y \to \mathcal{T}(\Sigma, X)$ slik at $l\sigma = t|_p$ og $t' = t[r\sigma]_p$* og
2. *det fins en type s slik at $t[x{:}s]_p$ er en velformet term samtidig som $l\sigma, r\sigma \in \mathcal{T}(\Sigma, X)_s$.*

Når det ikke er *helt* nødvendig å eksplisitt nevne variabelmengdene vil jeg i denne innføringen skrive $\underset{\mathcal{R}}{\longrightarrow}$ for $\underset{\mathcal{R}}{\longrightarrow}_X$.

4.2 Typeminskende omskrivningssystemer (1)

Ovenstående krav 2 til omskrivningsrelasjoner gjør (som vi skal se mange eksempler på) det vanskelig å resonnere om omskrivningssystemer. Det hadde i mange tilfeller vært ønskelig med et krav til f. eks. regelmengden som er enklere å beregne enn krav 2 men som sikrer "kontekstapplikasjon". Vi kaller derfor en regel $(\forall Y)\ l \longrightarrow r$ *kompatibel* dersom bruk av denne regelen gjør at krav 2 til omskrivningsrelasjonen alltid er oppfylt[2]. Mer formelt er $(\forall Y)\ l \longrightarrow r$ kompatibel hvis krav 2 til omskrivningsrelasjonen er tilfredsstilt for enhver term $t \in \mathcal{T}(\Sigma, X)$ og substitusjon $\sigma : Y \to \mathcal{T}(\Sigma, X)$ der $l\sigma = t|_p$.

Et kriterium sikrer kompatibilitet er *typeminskning* av reglene. En regel $l \longrightarrow r$ er typeminskende dersom $LS(l\sigma) \geq LS(r\sigma)$ for alle instanser $l\sigma \longrightarrow r\sigma$ av regelen. Typen s i punkt 2 i definisjon 4.3 kan da være lik $LS(l\sigma)$ siden $LS(r\sigma) \leq LS(r\sigma)$ impliserer $r\sigma \in \mathcal{T}(\Sigma, X)_{LS(l\sigma)}$. Et omskrivningssystem er typeminskende når enhver regel i systemet er det. En typeminskende regel $l \longrightarrow r$ er *typebevarende* når $LS(l\sigma) = LS(r\sigma)$ for alle instanser $l\sigma \longrightarrow r\sigma$ av regelen. En regel sies å være *ekte typeminskende* dersom den er typeminskende men ikke typebevarende, mens en regel*instans* $l\sigma \longrightarrow r\sigma$ er ekte typeminskende dersom $LS(r\sigma) < LS(l\sigma)$. En regel har imidlertid (oftest) uendelige mange instanser, slik at det er nødvendig å finne et enklere kriterium for typeminskning. Det er ikke tilstrekkelig at sjekke $LS(l)$ og $LS(r)$, men det er nok å sjekke $l \longrightarrow r$ for alle typer termer variable kan instansieres med.

[2] Med utypet tankegang betyr dette at en omskrivning alltid kan foretas dersom en regel matcher.

Definisjon 4.4 (Spesialisering) *En* spesialisering $\mu : X \rightarrow X'$ *er en variabeltilordning* $\mu = \{x_1 : s_1 \mapsto x_1 : s'_1, \ldots, x_n : s_n \mapsto x_n : s'_n\}$ *der* $s'_i \leq s_i$ *for alle* $i \in 1..n$ *og* $X = \bigcup_{i \in 1..n} x_i$. *Den homomorfe utvidelsen* $\mu^* : \mathcal{T}(\Sigma, X) \rightarrow \mathcal{T}(\Sigma, X')$ *av* μ *skrives også* μ.

Jeg bruker stort sett samme navn på variable i spesialiseringer selv om de tilhører ulike variabelmengder. En spesialisering gir kort fortalt en variabel en mindre (eller like stor) type.

Eksempel 4.3 Anta at $(S, \leq) = s'' < s' < s$ og $X = \{x_1 : s,\ x_2 : s',\ x_3 : s''\}$. Da er $\mu = \{x_1 \mapsto x_1 : s'',\ x_2 \mapsto x_2 : s'',\ x_3 \mapsto x_3 : s''\}$ og $\mu' = \{x_1 \mapsto x_1 : s',\ x_2 \mapsto x_2 : s',\ x_3 \mapsto x_3 : s''\}$ to spesialiseringer av X. Variabelen x_1 kan gis typene s, s' eller s'', x_2 typene s' eller s'' mens x_3 må beholde typen s''. Det fins følgelig 6 spesialiseringer av X.

En endelig variabelmengde har et endelig antall spesialiseringer sålenge enhver type har endelige mange typer mindre enn seg.

Teorem 4.5 ([GKK90]) *Et omskrivningssystem* $\mathcal{R}$ *er typeminskende dersom*

$$LS(l\mu) \geq LS(r\mu)$$

for enhver regel $l \longrightarrow r$ *i* $\mathcal{R}$ *og enhver spesialisering* μ *av variable i* l.

Jeg kaller $l\mu \longrightarrow r\mu$ for en *regelspesialisering* når $l \longrightarrow r$ er en regel og μ er en spesialisering av variable i l. For endelig systemer er altså typeminskning greit beregnbart.

Eksempel 4.4 La en signatur være gitt

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s'$
	$g : s \rightarrow s'$
	$a : s$

I denne signaturen ville en regel $(\forall x : s)\ f(x) \longrightarrow g(x)$ vært typeminskende, mens en regel $(\forall x : s)\ f(x) \longrightarrow a$ *ikke* ville vært det, fordi dens spesialisering $(\forall x : s')\ f(x) \longrightarrow a$ er typeøkende.

Typeminskning synes å være et forholdsvis restriktivt krav. I eksempelet over ville faktisk *alle mulige* regler vært kompatible, men typeminskningskravet diskvalifiserer flere av disse. Det kan derfor synes litt underlig når både Gnaedig, Kirchner og Kirchner i [Gn92a] (som *ikke* dreier seg om komplettering), [GKK90] og [GKK87] og Waldmann i [Wal92] alle antar at regelmengdene er typeminskende, noe jeg også vil gjøre i del II av denne oppgaven.

Man kan spørre hvor restriktivt typeminskningskravet egentlig er. Gnaedig et al. påstår at det viser seg i praksis at regler i et omskrivningssystem (eller likninger under komplettering) gjerne er (evt. lar seg orientere slik at de blir) typeminskende. Følgende eksempel viser at det av og til kan dukke opp systemer som opplagt vil føre til typeøkende regler:

Eksempel 4.5 (av Lars With, sitert i [CH93]) Den naturlige måten å definere en funksjon *square* (som ganger et tall med seg selv) på (gitt spesifikasjonen for Int og Nat og definisjonen av $*$) er:

Typer: Int, Nat
$Nat \leq Int$
Funksjoner: $* : Int, Int \rightarrow Int$
$square : Int \rightarrow Nat$
Regel: $(\forall x : Int)\ square(x) \longrightarrow x * x$

Regelen er typeøkende fordi $square(x)$ er av typen Nat mens $x * x$ har minste type Int.

Det er ikke utenkelig at man kan stille andre krav til omskrivningssystemer som sikrer kompatibilitet, men jeg har ikke sett noe eksempel på slike krav og det ligger utenfor denne oppgavens intensjoner å gjøre noe med dette.

En videre diskusjon om nødvendigheten av – og begrensningen til – typeminskning følger i neste kapittel.

4.3 Konfluens og konvergens

Det er litt mer komplisert å avgjøre konfluens i typeordnede enn i utypede systemer. Generelt må man se på alle spesialiseringer til variablene.

Eksempel 4.6 Systemet

Typer: s, s_1, s_2
$s_1 \leq s$
$s_2 \leq s$
Funksjoner: $f : s_1 \rightarrow s$
$f : s_2 \rightarrow s$
$a : s_1$
$b : s_2$
Regler: $(\forall x : s_1)\ f(x) \longrightarrow a$
$(\forall y : s_2)\ f(y) \longrightarrow b$

med regelmengden $\mathcal{R}$ er konfluent. Anta så at signaturen inneholdt en type $s_{12} \leq s_1, s_2$ og en konstant $c : s_{12}$. Da ville ikke systemet vært konfluent fordi $f(c) \xrightarrow[\mathcal{R}]{} a$ og $f(c) \xrightarrow[\mathcal{R}]{} b$.

Et første eksempel på nødvendigheten av typeminskning blir illustrert av at det nesten er umulig å avgjøre konfluens for ikke-typeminskende systemer.

Eksempel 4.7 La oss se på følgende signatur og ikke-typeminskende regelmengde $\mathcal{R}$:

Typer: s, s'
$s' \leq s$
Funksjoner: $f : s' \rightarrow s$
$a, b, c, e : s'$
$d : s$
Regler: $a \longrightarrow b$
$a \longrightarrow c$
$b \longrightarrow e$
$c \longrightarrow d$
$d \longrightarrow e$

Systemet kan ved første øyekast se konfluent ut, men termen $f(a)$ kan omskrives $f(a) \xrightarrow[\mathcal{R}]{} f(b) \xrightarrow[\mathcal{R}]{} f(e)$ og $f(a) \xrightarrow[\mathcal{R}]{} f(c)$. Termen $f(c)$ kan ikke omskrives videre fordi $f(d)$ ikke er en velformet term, slik at $\mathcal{R}$ ikke er konfluent fordi $f(a)$ har to normalformer.

Dette var en liten smaksprøve på vanskelighetene med å avgjøre konfluens. Dersom et system $\mathcal{R}$ er konfluent og terminerende er det konvergent, som også i typeordnet omskrivning betyr at $\xleftrightarrow[\mathcal{R}]{*}$-like termer har like normalformer.

4.4 Retrakter

Jeg har ikke sett noen beskrivelse av retrakter i forbindelse med typeordnet omskrivning. Grunnen er sannsynligvis at retrakter forlanger "run-time-sjekk", slik at retrakter naturlig faller inn under betingede omskrivningssystemer. Dette kan virke noe ubehagelig da motivasjonen for å jobbe med typeordnede spesifikasjoner og omskrivningssystemer nettopp er å unngå betingede likninger og regler. Jeg synes likevel ikke at behovet av retrakter diskvalifiserer typeordnet omskrivning. For det første er betingelsen i en retraktregel alltid enkelt beregnbar i motsetning til en generell betingelse i et betinget omskrivningssystem. For det andre fins det mange typeordnede omskrivningssystemer som *ikke* kan gjøres om til ubetingede utypede systemer, men der man aldri vil ha behov for retrakter. Det modifiserte kaffebønnespillet i eks. 4.1 er et eksempel på et slikt system. Jeg mener derfor at retrakter må tillates i typeordnede omskrivningssystemer for å gi dem full anvendbarhet. Jeg ville anta at retrakter brukes på en måte følgende eksempel antyder.

Eksempel 4.8 La oss representere fakultetsfunksjonen ved et omskrivningssystem $\mathcal{R}$, der vi bruker operatorene[3] 0, s og p som i eksempel 6.1 for å bygge opp typene $NzNat$, Nat og Int der $NzNat \leq Nat \leq Int$. Noen av de interessante deklarasjonene som kommer i tillegg til deklarasjonene i eks. 6.1 er da

$$\begin{aligned} p &: NzNat \rightarrow Nat \\ - &: Int, Int \rightarrow Int \\ * &: Int, Int \rightarrow Int \\ * &: NzNat, NzNat \rightarrow NzNat \\ ! &: Nat \rightarrow NzNat \end{aligned}$$

[3] uten parenteser rundt argumentet til s og p

der den andre deklarasjonen av $*$ er nødvendig for å sikre typeminskning av $\mathcal{R}$ og p er forgjenger-funksjonen som brukes $(\forall x : Nat)\ psx \longrightarrow x$. Den naturlige definisjonen av fakultetsfunksjonen ville være

$$\begin{array}{rcl} (\forall x : NzNat)\ x! & \longrightarrow & x * ((px)!) \\ 0! & \longrightarrow & s0 \end{array}$$

Jeg ser ikke mer på definisjonen av andre funksjonssymboler.

Det er opplagt at uttrykket $(3-1)!$ har mening i matematikk og burde kunne beregnes mens $(1-3)!$ ikke har det. Anta at vi ønsker å beregne $(3-1)!$, dvs. $(sss0 - s0)!$. Dessverre kan ingen omskrivning starte med denne "termen", siden $sss0 - s0$ er av typen Int slik at $(sss0 - s0)!$ ikke er en velformet term. Man kan da innføre en retrakt-funksjon

$$r : Int \rightarrow Nat$$

Denne må brukes på følgende måte:

$$\begin{array}{rl} \textbf{if} <x \text{ er av typen } Nat> \textbf{ then} & r(x) \longrightarrow x \\ \textbf{if} <x \text{ irredusibel og ikke av type } Nat> \textbf{ then} & r(x) \longrightarrow \textbf{abort}(x \text{ "ikke av type Nat"}) \end{array}$$

Termen $r(sss0 - s0)$ har nå typen Nat, slik at $(r(sss0 - s0))!$ er veldefinert. $(3-1)!$ kan nå beregnes

$$(r(sss0 - s0))! \xrightarrow[\mathcal{R}]{+} (r(ss0))! \xrightarrow[\mathcal{R}]{r(x)\rightarrow x} ss0! \xrightarrow[\mathcal{R}]{+} ss0$$

Legg merke til at typedeklarasjonene i dette tilfellet sikrer at innholdet i retrakten blir beregnet først, selv om dette normalt ikke har noen betydning.

Optimistisk forsøker vi nå beregne $(1-3)!$ på samme måten og får omskrivningssekvensen

$$(r(s0 - sss0))! \xrightarrow[\mathcal{R}]{+} (r(pp0))! \xrightarrow[\mathcal{R}]{} \textbf{abort}(\text{"}pp0 \text{ ikke av type } Nat\text{"})$$

der vi heller ikke nå har valgfrihet i regelbruken en eneste gang. Merk at retrakter kan legges til automatisk til uttrykk som $(ss0 - s0)!$ uten at en "bruker" trenger å bekymre seg om dette.

Kan $(3-1)!$ beregnes mer elegant i mangetypet eller utypet omskrivning?

Betingede omskrivningssystemer ligger langt utenfor skopet til både denne oppgaven og de fleste artiklene om typeordnet omskrivning. Såvidt jeg forstår forskes det idag endel på å klare å beregne uttrykk som $(3-1)!$ uten måtte bruke betinget omskrivning.

4.5 Oppsummering

Dette kapitlet har gitt en kort innføring i det største (og eneste?) problemet i typeordnet omskrivning, nemlig det at kongruensdefinisjonen og muligheten av å få ikke-velformede termer gjør at omskrivningssystemer ikke er kompatible. Ved å forlange typeminskning forsvinner disse problemene.

Jeg vil være nødt til å anta typeminskning for å vise typeordnet terminering.

Kapittel 5

Typeordnet komplettering

Typeordnet omskrivning brukes ofte til å lage desisjonsprosedyrer for gyldighetsproblemet i typeordnede spesifikasjoner. Selv om komplettering faller utenom det egentlige temaet til denne oppgaven kan det likevel være interessant å peke på noen fallgruver i typeordnet komplettering og se på andre forskjeller fra utypet komplettering.

Som vi allerede har sett, har mange problemer en tendens til å forsvinne med én gang man forlanger at reglene skal være typeminskende. Slik er det også i komplettering, noe som gir en enda sterkere begrunnelse på hvorfor det er tilstrekkelig å jobbe med systemer som innehar denne egenskapen.

Grovt sett er forskjellen fra utypet komplettering at dennes to grunnpillarer, lemmaet om kritiske par og Birkhoffs teorem, *ikke* nødvendigvis gjelder i ikke-typeminskende systemer og at man kan dedusere "inkonsistente" regler dersom man ikke er forsiktig med variabelmengdene når signaturen kan inneholde tomme typer. Av denne grunn er variabelmengdene viktige.

Typeordnet komplettering foregår ellers etter de samme prinsippene som i den utypede verdenen. Man starter med en gitt algebraisk spesifikasjon og definerer en termineringsordning $\succ$ og prøver å orientere likningene til regler. Deretter forsøker man å gjøre regelmengden konfluent ved å undersøke alle mulige kritiske par og orientere disse slik at terminering, typeminskning og ($\approx$-)sunnhet er invariant. Dersom man ender opp med en (endelig) regelmengde der begge elementene i et kritisk par har samme normalformer står man igjen med en desisjonsprosedyre for typeordnet gyldighet. En likning $(\forall X)\ t \approx t'$ er gyldig i alle $\langle \Sigma, E \rangle$-algebraer hvis og bare hvis termene har samme normalform.

Jeg vil som alltid når jeg jobber med likninger over termer anta at signaturene er koherente.

5.1 Nødvendighet av typeminskning – Birkhoffs teorem og kritiske par

I dette kapitlet vil jeg også se på regler som likninger og omvendt slik at likningsmengden $E_{\mathcal{R}} = \{(\forall X)\ l \approx r | (\forall X)\ l \longrightarrow r \in \mathcal{R}\}$ og regelmengden $\mathcal{R}_E$ er gitt $E = \{(\forall X)\ v \longrightarrow h | (\forall X)\ v \approx h \in E \vee (\forall X)\ h \approx v \in E\}$. Av fremstillingstekniske grunner skriver jeg $\xrightarrow[E]{}$ for $\xrightarrow[\mathcal{R}_E]{}$ og $\approx_{\mathcal{R}}$ for $\approx_{E_{\mathcal{R}}}$.

Knuth-Bendix-komplettering bygger som nevnt i stor grad på lemmaet som sier at omskrivningssystemer $\mathcal{R}$ er konfluente dersom alle kritiske par er inneholdt i relasjonen $\xrightarrow[\mathcal{R}]{*} \circ \xleftarrow[\mathcal{R}]{*}$ og på Birkhoffs teorem. Ingen av disse resultatene holder i typeordnet algebra dersom regelmengden ikke er typeminskende.

Smolkas eksempel viser at Birkhoffs teorem ikke gjelder for typeordnede omskrivningssystemer som ikke er lukket under kontekstapplikasjon.

Eksempel 5.1 (Smolka, sitert i [GKK90]) La en koherent signatur Σ og en konfluent regelmengde $\mathcal{R}$ være gitt som følger:

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s' \rightarrow s$
	$a : s'$
	$b : s'$
	$c : s$
Regler:	$a \longrightarrow c$
	$b \longrightarrow c$

Likningen $a \approx b$ er gyldig i alle algebraer som tilfredsstiller $E_{\mathcal{R}}$. Kongruens fører til at $f(a) \approx f(b)$ er gyldig, samtidig som vi ikke har $f(a) \xleftrightarrow[\mathcal{R}]{*} f(b)$ fordi ingen regel kan brukes på $f(a)$ siden $f(c)$ ikke er en velformet term.

En kompletteringsprosedyre bør altså bare bruke kompatible regler. Av kravene til kompatibilitet later det til at det bare er typeminskningskravet som også sikrer at unifikasjon aldri gir uendelige mange mest generelle unifikatorer.

Teorem 5.1 (Birkhoffs teorem for typeordnede systemer) *Anta at Σ er en koherent typeordnet signatur og at $\mathcal{R}$ er et typeminskende og konfluent omskrivningssystem. Da gjelder*

$$(\forall X)\, t \approx_{\mathcal{R}} t' \iff t \xleftrightarrow[\mathcal{R}]{*}_X t'$$

Bevis: Følger rett fra teoremene 3.10 og 4.11 i [GKK90]. □

Det ekstra kravet om konfluens kan se merkelig ut men regelmengden $\{c \longrightarrow a,\ c \longrightarrow b\}$ fra signaturen i eksempel 5.1 er typeminskende samtidig som vi heller ikke her har $f(a) \xleftrightarrow[\mathcal{R}]{*} f(b)$.

Dette er altså én av grunnene til at man stort sett bare resonnerer om typeminskende regler under komplettering. Den andre hovedgrunnen er de kritiske parene. Eksemplet nedenfor (sitert i [CH93]) viser (på en litt annen måte enn eksempel 4.7) at et ikke-typeminskende system ikke er konfluent selv om det ikke har venstresider av regler som overlapper i ikke-variabel-posisjoner.

Eksempel 5.2 La $\langle \Sigma, \mathcal{R} \rangle$ være

Typer: s, s'
$s' \leq s$
Funksjoner: $f : s \longrightarrow s$
$a : s'$
$b : s$
Regler: $a \longrightarrow b$
$(\forall x : s')\ \ f(x) \longrightarrow x$

Det er ingen overlappinger i venstresidenes ikke-variabel-posisjoner ("kritiske par"), men systemet er likevel ikke konfluent fordi termen $f(a)$ kan omskrives enten $f(a) \xrightarrow[\mathcal{R}]{} a \xrightarrow[\mathcal{R}]{} b$ eller $f(a) \xrightarrow[\mathcal{R}]{} f(b)$, der både b og $f(b)$ er irredusible termer.

Knuth og Bendix' lemma om kritiske par [KB70] holder når regelmengden er typeminskende[1]. Eksempel 5.2 er også meget interessant fordi $\mathcal{R}$ faktisk er kompatibel. I vår jakt på andre krav enn typeminskning holder det dermed faktisk *ikke* å sikre at $\mathcal{R}$ er kompatibel.

For å unngå disse problemene vil jeg i resten av kapitlet forlange at reglene i en kompletteringsprosedyre er typeminskende.

5.2 Kritiske par

Gitt en likningsmengde E går kompletteringens neste fase ut på å lage en regelmengde $\mathcal{R}$ slik at $\approx_E = \approx_{\mathcal{R}}$. Regelmengden $\mathcal{R}$ er da sannsynligvis ikke konfluent og den neste delen av arbeidet er å gjøre $\mathcal{R}$ til en konfluent, terminerende og typeminskende regelmengde $\mathcal{R}'$ slik at $\approx_{\mathcal{R}} = \approx_{\mathcal{R}'}$. Dette gir $\approx_E = \ \approx_{\mathcal{R}'} \ = \xleftrightarrow[\mathcal{R}']{*} = \xrightarrow[\mathcal{R}']{!} \circ \xleftarrow[\mathcal{R}']{!}$ og E-gyldighetsproblemet er dermed løst.

For å få et omskrivningssystem $\mathcal{R}$ konfluent går man frem som i utypet komplettering. Man undersøker om det fins to (ikke nødvendigvis disjunkte) regler $l \longrightarrow r$ og $v \longrightarrow h$ i $\mathcal{R}$ (der vi omnavner variable slik at $\mathcal{V}(l) \cap \mathcal{V}(v) = \emptyset$) som "overlapper" i en ikke-variabel-posisjon, dvs. om det fins en substitusjon σ og posisjon p der $v|_p \notin X$ slik at

$$l\sigma = (v|_p)\sigma$$

Spørsmålet hvorvidt det fins en slik substitusjon σ kalles *unifikajsonsproblemet*. Det fins imidlertid uendelige mange slike σ dersom unifikasjonen har en løsning, så vi er bare interessert i mengden Φ av *mest generelle unifikatorer*, der Φ er den minste mengden av unifikatorer som oppfyller

$$\forall \sigma \mid l\sigma = (v|_p)\sigma \Longrightarrow \exists \rho \in \Phi | \exists \phi | \sigma = \rho\phi$$

for substitusjoner σ, ρ og ϕ. Når det fins en slik unifikator ρ, kan termen $v\rho$ skrives om til enten $h\rho$ eller $v\rho[r\rho]_p$ (som er lik $(v[r]_p)\rho$) med bruk av hhv. regel $v \longrightarrow h$ eller $l \longrightarrow r$. Paret $\langle h\rho, v\rho[r\rho]_p \rangle$ kalles da et *kritisk par* for $\mathcal{R}$, og dersom paret ikke er inneholdt i $\xrightarrow[\mathcal{R}]{*} \circ \xleftarrow[\mathcal{R}]{*}$ så gjør dette paret at $\mathcal{R}$ ikke er konfluent. For å prøve å gjøre $\mathcal{R}$ konfluent orienterer man paret og legger det til regelmengden på en mer eller mindre innfløkt måte. Når så alle de kritiske parene i (den utvidete) $\mathcal{R}$ er undersøkt og er med i $\xrightarrow[\mathcal{R}]{*} \circ \xleftarrow[\mathcal{R}]{*}$ har vi fått den ønskede regelmengden $\mathcal{R}'$ som kan brukes som desisjonsprosedyre for E-gyldighetsproblemet.

[1] Dette er en direkte konsekvens av teorem 5.5.

5.2.1 Typeordnet unifikasjon

I utypede systemer er det forholdsvis enkelt å finne en eventuell mest generell unifikator for to termer t og t' (se f. eks. [Kir92]). I typeordnet algebra kan man ikke bare gå rett frem, men må til enhver tid ha kjennskap til hele signaturen. Følgende enkle eksempel gir et lite inntrykk av noen av vanskelighetene med typeordnet unifikasjon.

Eksempel 5.3 La en koherent signatur være

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s, s \rightarrow s$
	$f : s, s' \rightarrow s'$
	$f : s', s \rightarrow s'$
	$f : s', s' \rightarrow s'$
	$g : s, s, s \rightarrow s$
	$a : ?$
	$b : ?$
Variable:	$x : s',\ y : s,\ z : s$

Anta at vi prøver å løse[2]

$$g(x, y, z) \stackrel{?}{=} g(f(y, z), a, b)$$

Denne må på vanlig måte dekomponeres til

$$x \stackrel{?}{=} f(y, z) \qquad y \stackrel{?}{=} a \qquad z \stackrel{?}{=} b$$

I utypet unifikasjon løses disse trivielt, men her ligger problemet i at $f(y, z)$ har typen s, så det kan se ut som om unifikasjonen ikke har en løsning. La så konstanten a ha typen s'. Da er $\{y \mapsto a, x \mapsto f(a, z)\}$ en unifikator fordi $f(a, z)$ er av typen s'. Har både a og b type s går ikke unifikasjonen opp. Det er lett å forestille seg hvor komplisert bildet hadde vært dersom a og b var erstattet med større termer.

Typeordnet unifikasjon er såpass vanskelig at det har blitt et eget forskningsområde det er blitt gjort ganske mye på. I en innføring er følgende proposisjon tilstrekkelig:

Proposition 5.2 *Gitt en endelig regulær signatur Σ har to termer i $\mathcal{T}(\Sigma, X)$ maksimalt et endelig antall mest generelle unifikatorer, og disse er effektivt beregnbare.*

Bevis: Følger rett fra teorem 4.6 i [Wal92]. □

5.2.2 Kritiske par

Vi kan nå definere kritiske par, der variabelmengden er det eneste nye i forhold til den utypede varianten.

[2] der $\stackrel{?}{=}$ står for "er unifiserbar med?"

Definisjon 5.3 (Kritisk par (Jouannaud og Kirchner)) *La* $(\forall X)\ v \longrightarrow h$ *og* $(\forall X')\ l \longrightarrow r$ *være to regler slik at* $X \cap X' = \emptyset$ *og* Φ *er mengden av mest generelle unifikatorer til* l *og* $v|_p$ *der* $v|_p \notin X$. *For enhver* $\rho \in \Phi$ *er likningen*

$$(\forall\ \mathcal{V}(v\rho))\ (v[r]_p)\rho \approx h\rho$$

et kritisk par *til regelen* $(\forall X')\ l \longrightarrow r$ *over posisjonen* p *i regelen* $(\forall X)\ v \longrightarrow h$.

5.3 Kompletteringsprosedyren

Kompletteringsprosedyren kan som i det utypede tilfellet beskrives ved et inferenssystem der datastrukturen består av en mengde likninger E og en mengde regler $\mathcal{R}$. Inferensreglene er vist i figur 5.1 der $\succ$ er en termineringsordning.

Orient:	$\dfrac{E \cup \{(\forall X)\ p \approx q\}, \mathcal{R}}{E, \mathcal{R} \cup \{(\forall X)\ p \longrightarrow q\}}$	$p \succ q$ og $(\forall X)\ p \longrightarrow q$ typeminskende
Deduce:	$\dfrac{E, \mathcal{R}}{E \cup \{(\forall X)\ p \approx q\}, \mathcal{R}}$	$(\forall X)\ p \approx q$ kritisk par i $\mathcal{R}$
Simplify:	$\dfrac{E \cup \{(\forall X)\ p \approx q\}, \mathcal{R}}{E \cup \{(\forall X)\ p' \approx q\}, \mathcal{R}}$	hvis $p \xrightarrow[\mathcal{R}]{} p'$
Delete:	$\dfrac{E \cup \{(\forall X)\ p \approx p\}, \mathcal{R}}{E, \mathcal{R}}$	
Compose:	$\dfrac{E, \mathcal{R} \cup \{(\forall X)\ l \longrightarrow r\}}{E, \mathcal{R} \cup \{(\forall X)\ l \longrightarrow r'\}}$	hvis $r \xrightarrow[\mathcal{R}]{} r'$
Collapse:	$\dfrac{E, \mathcal{R} \cup \{(\forall X)\ l \longrightarrow r\}}{E \cup \{(\forall X)\ l' \approx r\}, \mathcal{R}}$	hvis $l \xrightarrow[\mathcal{R}]{v \to h} l'$ og $l >> v$

Figur 5.1: *Inferensregler for typeordnet komplettering.*

Relasjonen $>>$ defineres slik at

$$l >> v \iff \exists p, \sigma | v\sigma = l|_p \wedge p \neq \epsilon \text{ eller } v\sigma = l \text{ og } \sigma \text{ ikke er en variabelomnavning}$$

Av notasjonsmessige grunner har jeg i reglene *simplify* og *orient* bare én retning av likningen $p \approx q$. Reglene gjelder også for den omvendte $q \approx p$.

Inferensreglene ser altså ut som inferensreglene for utypet komplettering. Den eneste forskjellen ligger i typeminskningskravet til *orient*. Det synes som om vanskelighetene i typeordnet komplettering ligger skjult i typeminskningskravet og i typeordnet unifisering.

5.3.1 Korrekthet av inferenssystemet

Leo Bachmair innførte i sin doktorsavhandling en meget interessant måte å resonnere om korrekthet av inferenssystemer for komplettering på.

La oss starte kompletteringen med likningsmengden $E_0 = E$ og en (oftest) tom regelmengde $\mathcal{R}_0$. Enhver bruk av en inferensregel på paret $\langle E_i, \mathcal{R}_i \rangle$ gir paret $\langle E_{i+1}, \mathcal{R}_{i+1} \rangle$. I korthet går så metoden til Bachmair ut på å vise at dersom det fins et *bevis* for $t \approx t'$ i $\langle E_i, \mathcal{R}_i \rangle$ så fins det også et (i en viss forstand "mindre") bevis for $t \approx t'$ i $\langle E_{i+1}, \mathcal{R}_{i+1} \rangle$, der et bevis i $\langle E, \mathcal{R} \rangle$ er en sekvens av termer $t_1, t_2, \ldots, t_n$ slik at $t = t_1$, $t' = t_n$ og for alle $j < n$ så gjelder enten $t_j \xrightarrow[\mathcal{R}]{} t_{j+1}$, $t_j \xleftarrow[\mathcal{R}]{} t_{j+1}$ eller $t_j \xleftrightarrow[E]{} t_{j+1}$. Målet med inferenssystemet er å komme frem til en regelmengde $\mathcal{R}'$ slik at alle likninger $t_1 \approx t_n$ som er logiske konsekvenser av E skal ha et bevis på formen

$$t_1 \xrightarrow[\mathcal{R}']{*} t_j \xleftarrow[\mathcal{R}']{*} t_n$$

I den utypede verdenen er dette greit fordi $\approx_{E_i \cup R_i} = \xleftrightarrow[E_i \cup R_i]{*}$ og $\xleftrightarrow[E_i \cup R_i]{*} = \xleftrightarrow[E_{i+1} \cup R_{i+1}]{*}$ holder for "normale" inferenssystemer. Ingen av disse behagelige egenskapene holder i typeordnet komplettering. La E_0 være likningene $\{a \approx c, b \approx c\}$ i eksempel 5.1 og vi viste der at $f(a) \approx_{E_0} f(b)$ men ikke $f(a) \xleftrightarrow[E_0]{*} f(b)$. La oss så orientere likningene i E_0 slik at $\mathcal{R}_2 = \{c \longrightarrow a, c \longrightarrow b\}$ og ved å dedusere og orientere det kritiske paret $a \approx b$ får vi $\mathcal{R}_4 = \{c \longrightarrow a, c \longrightarrow b, a \longrightarrow b\}$. Dermed holder ikke $\xleftrightarrow[E_i \cup \mathcal{R}_i]{*} = \xleftrightarrow[E_{i+1} \cup \mathcal{R}_{i+1}]{*}$ fordi $f(a) \xleftrightarrow[E_0 \cup \mathcal{R}_0]{*} f(b)$ *ikke* holder mens $f(a) \xleftrightarrow[E_4 \cup \mathcal{R}_4]{*} f(b)$ holder. Den eneste form for sunnhet som blir bevart av inferensreglene er $\approx_{E_i \cup \mathcal{R}_i} = \approx_{E_{i+1} \cup \mathcal{R}_{i+1}}$.

Man pleier å si at resultatet av en sekvens $\langle E_0, \mathcal{R}_0 \rangle, \langle E_1, \mathcal{R}_1 \rangle, \ldots$ er paret $\langle E_\infty, \mathcal{R}_\infty \rangle$ hvor $E_\infty = \bigcup_i \bigcap_{j \geq i} E_j$ og $\mathcal{R}_\infty = \bigcup_i \bigcap_{j \geq i} \mathcal{R}_j$ og der i går fra 0 til n dersom kompletteringen avsluttes med paret $\langle E_n, \mathcal{R}_n \rangle$ og fra 0 til ∞ dersom en slik n ikke finnes. Det er mange måter å bruke inferensreglene på; man kan for eksempel dedusere samme kritiske par hele tiden. Derfor forlanger man at bruken av inferensreglene skal være fornuftig, eller *fair*[3].

Definisjon 5.4 (Fair komplettering [GKK90]) *En komplettering er* fair *dersom* $E_\infty = \emptyset$ *og alle kritiske par i* $\mathcal{R}_\infty$ *er beregnet.*

Da skulle alt ligge til rette for korrekthetsbevis av kompletteringsprosedyren. Her må vi huske på variabelmengdene i omskrivningsrelasjonen $\xrightarrow[\mathcal{R}]{}_X$, der jeg kaller $\xrightarrow[\mathcal{R}]{}_X$-normalformen til en term $t \in \mathcal{T}(\Sigma, X)$ for $t!\mathcal{R}_X$.

Teorem 5.5 (Korrekthet) *Anta gitt en koherent signatur* Σ*, en typeordnet likningsmengde* E *og en typeordnet variabelmengde* X*. Dersom kompletteringen som starter med* E *er fair og* $\mathcal{R}_\infty$ *er terminerende så er*

$$t!\mathcal{R}_{\infty X} = t'!\mathcal{R}_{\infty X}$$

hvis og bare hvis

$$(\forall X)\ t \approx_E t'$$

er en E*-gyldig likning*

[3] En videre filosofering rundt dette begrepet kan man finne i f. eks. [Lin92].

Bevis: Av notasjonsmessige grunner bruker jeg i beviset relasjonssymbolet $\approx_E^X$ der $\langle t, t' \rangle \in \approx_E^X \iff (\forall X)\, t \approx_E t'$.

Man kan først legge merke til at $\mathcal{R}_\infty$ er inneholdt i $\succ$ og er typeminskende.

Fordi $\approx_{E_i \cup \mathcal{R}_i}^X = \approx_{E_{i+1} \cup \mathcal{R}_{i+1}}^X$ over $\mathcal{T}(\Sigma, X)$ (følger fra lemma 7.5 og teorem 3.10 i [GKK90]) har vi $\approx_E^X = \approx_{E_0}^X = \approx_{\mathcal{R}_\infty}^X$ siden $R_0 = E_\infty = \emptyset$ når kompletteringen er fair. Teorem 7.8 i [GKK90] sier at $\xrightarrow[\mathcal{R}_\infty]{}_X$ er Church-Rosser, dvs. $\approx_{\mathcal{R}_\infty}^X = \xrightarrow[\mathcal{R}_\infty]{*}_X \circ \xleftarrow[\mathcal{R}_\infty]{*}_X$. Konvergens fører til éntydige normalformer som gir det ønskede resultatet. □

5.4 Tomme typer

Muligheten for at signaturen kan ha tomme typer introduserer en vanskelighet vi ikke finner i utypet komplettering. En type s i en signatur Σ er *tom* når det ikke finnes grunntermer av type s. Dersom man ikke er veldig forsiktig under komplettering, kan man dedusere inkonsistente regler, slik som i følgende eksempel til Goguen et al. sitert i [GKK90]:

Eksempel 5.4 La den koherente signaturen Σ og likningsmengden E være

Typer:	$Tom, Bool$
Funksjoner:	$f : Tom \rightarrow Bool$
	$T : Bool$
	$F : Bool$
Likninger:	$(\forall x : Tom)\ \ f(x) \approx T$
	$(\forall x : Tom)\ \ f(x) \approx F$

Likningene orienteres til $(\forall x : Tom)\ f(x) \longrightarrow T$ og $(\forall x : Tom)\ f(x) \longrightarrow F$. Her finnes en klar overlapp, og dersom man er uforsiktig kan man dedusere og orientere det kritiske paret $T \longrightarrow F$.

Typen Tom inneholder ikke noen grunntermer slik at mengden mengden $\mathcal{A}_{Tom}$ kan være $\emptyset$. En $\langle \Sigma, E \rangle$-algebra kan være algebraen $\mathcal{RE}$, der $\mathcal{RE}_{Tom}$ er mengden av alle rosa elefanter[4]. Funksjonssymbolet f kan tolkes av en funksjon $\mathcal{RE}_f^{Tom \rightarrow Bool}$ som "antall bein" og typen $Bool$ tolkes som de naturlige tallene. Til sist tolkes T som 2 og F som 8. Algebraen $\mathcal{RE}$ er en $\langle \Sigma, E \rangle$-algebra fordi alle rosa elefanter har 8 bein, siden det ikke finnes noen rosa elefanter som *ikke* har 8 bein. På samme måte er likningen $(\forall x : Tom)\ f(x) \approx T$ gyldig i $\mathcal{RE}$. Likningen "$T \approx F$" er imidlertid *ikke* gyldig i $\mathcal{RE}$ fordi 2 er forskjellig fra 8.

Eksempelet over kan synes underlig i lys av teorem 5.5, men ser man litt nøyere på definisjonen av kritiske par, oppdager man at det kritiske paret *ikke* er $T \approx F$, men $(\forall x : Tom)\ T \approx F$ (når unifikatoren ρ til $f(x' : Tom) \longrightarrow F$ og $f(x : Tom) \longrightarrow T$ er $\{x \mapsto x,\ x' \mapsto x\}$ er variabelmengden $\mathcal{V}(f(x)\rho) = \{x : Tom\}$). Likningen $(\forall x : Tom)\ T \approx F$ er gyldig i $\mathcal{RE}$ fordi skulle den ikke vært gyldig måtte det ha eksistert en rosa elefant slik at $2 \neq 8$, og en slik elefant finnes jo ikke[5]. Kompletteringsprosedyren gir da også riktig resultat.

[4] som naturligvis er tom (unntatt i "meget fuktige" øyeblikk)

[5] Det er vel et litt dypere spørsmål hvorvidt $\forall x \in \emptyset | 2 = 8$ virkelig kan sies å være et sant utsagn, men Gnaedig, Gougen, Meseguer et al. må ha ment det slik.

Eksempel (forts. av eks. 5.4) Kompletteringen ender opp med regelmengden

$$\mathcal{R}_\infty = \{(\forall x{:}Tom)\ f(x) \longrightarrow T,\ (\forall x{:}Tom)\ f(x) \longrightarrow F,\ (\forall x{:}Tom)\ T \longrightarrow F\}$$

Da har vi $T!\mathcal{R}_{\infty\{y:Tom\}} = F!\mathcal{R}_{\infty\{y:Tom\}}$ slik at likningen $(\forall y : Tom)\ T \approx F$ (ikke uventet) i følge teorem 5.5 er gyldig i alle $\langle\Sigma, E\rangle$-algebraer.

Merk også at det *ikke* fins en omskrivning $T \xrightarrow[\mathcal{R}_\infty]{}_{\emptyset} F$ fordi det ikke eksisterer en substitusjon $\sigma : \{x : Tom\} \rightarrow \mathcal{T}(\Sigma, \emptyset)$, siden x ikke kunnet blitt tilordnet en verdi. I omskrivningen $T \xrightarrow[\mathcal{R}_\infty]{}_{\{y:Tom\}} F$ er substitusjonen $\sigma : \{x : Tom\} \rightarrow \mathcal{T}(\Sigma, \{y : Tom\})$ lik $\{x \mapsto y\}$.

Dette er altså grunnen til at variabelmengdene er så viktige i typeordnet algebra, helt fra definisjonen av utledbarhet til "nedarving" av variabelmengder i kompletteringsprosedyrene.

Hvordan skal man så behandle tomme typer under komplettering? Den enkleste løsningen, strutsemetoden, valgt av bl. a. Waldmann [Wal92], er å se helt bort i fra tomme typer under komplettering ved å anta at enhver type i signaturen inneholder minst én grunnterm. På denne måten unngår han også kluss med variable i sine definisjoner. Svakheten kan være at denne fristende antagelsen diskvalifiserer parametriserte typer som man av og til må anta kan være tomme. Dersom leseren av denne innføringen velger samme infallsvinkel kan han/hun helt se bort i fra variabelmengder i f. eks. relasjonen $\xrightarrow[\mathcal{R}]{}_X$.

I steden for å anta tomme typer foreslår Gnaedig et al. [GKK90] å prøve å "huske" hvilke variable som "forsvinner" under komplettering. På disse områdene kan det være rom for nærmere granskning i sømmene for den meget interesserte.

5.5 Typeminskning – enda en gang

Vi har altså sett at typeminskningskravet gir omskrivningssystemer mange behagelige egenskaper, selv om eksemplene 4.4 og 4.5 også har vist at typeminskning kan være unødvendig strengt eller umulig å tilfredsstille.

Et problem som kan dukke opp under komplettering er også at en likning ikke kan orienteres slik at den er både typeminskende og termineringsbevarende.

Eksempel 5.5 Likningen i spesifikasjonen

Typer:	s, s', s''
	$s' \leq s$
	$s'' \leq s'$
Funksjoner:	$f : s \longrightarrow s'$
	$f : s' \longrightarrow s''$
	$a : s$
Likning:	$(\forall x{:}s)\ f(f(x)) \approx f(x)$

kan ikke orienteres fordi regelen $(\forall x{:}s)\ f(f(x)) \longrightarrow f(x)$ er typeøkende mens regelen $(\forall x{:}s)\ f(x) \longrightarrow f(f(x))$ gir opphav til en uendelig omskrivningssekvens

$$f(a) \longrightarrow f(f(a)) \longrightarrow f(f(f(a))) \longrightarrow \cdots$$

Et siste eksempel viser at en likning av og til ikke kan orienteres typeminskende i det hele tatt:

Eksempel 5.6 Hvis situasjonen

Typer:	s, s_1, s_2
	$s_1 \leq s$
	$s_2 \leq s$
Funksjoner:	$f : s_1 \rightarrow s$
	$a : s$
	$b : s_1$
	$c : s_2$
Regler:	$a \longrightarrow b$
	$a \longrightarrow c$

oppstår under komplettering, dukker det kritiske paret $b \approx c$ opp, og denne kan ikke orienteres på typeminskende måte fordi s_1 og s_2 er usammenlignbare.

Ikke uventet forskes det nå på andre idéer som løser slike problemer, og da særlig med tanke på komplettering.

Idéen til Chen og Hsiang [CH93] bygger på at også ikke velformede termer har en mening, i hvert fall som "mellomledd" i en avledning. Hvis $f(b)$ i eksempel 5.6 har mening og $b \approx c$ så burde også den ikke velformede termen $f(c)$ ha en tilsvarende mening (i enhver algebra). Denne fremgangsmåten ender opp i et *typekonvergenskrav* til regelmengden, som sier at for enhver regel $l \longrightarrow r$ i $\mathcal{R}$ og enhver substitusjon σ så skal det finnes en velformet term u slik at $l\sigma \xrightarrow[\mathcal{R}]{*} u \xleftarrow[\mathcal{R}]{*} r\sigma$[6]. Typekonvergens kan være en interessant infallsvinkel, men siden det ikke er helt trivielt å fastslå denne egenskapen velger jeg å ikke jobbe videre med slike systemer.

Gnaedig, Kirchner og Kirchner [GKK90] har en litt annen tanke om dette. Tar vi igjen utgangspunkt i eksempel 5.6, så er $b \approx c$. I enhver tolkende algebra $\mathcal{A}$ må $\mathcal{A}_b^{s_1} = \mathcal{A}_c^{s_2}$, som betyr at dette elementet er med i $\mathcal{A}_{s_1} \cap \mathcal{A}_{s_2}$. Dette faktum kan man så benytte på mange måter. I [GKK90] blir det innført en ny type $s_1 \diamond s_2$ (kan passelig leses "s_1 og s_2" eller "s_1 snitt s_2") slik at $s_1 \diamond s_2 \leq s_1, s_2$. Denne typen kan i sin tur brukes på forskjellige måter. Gnaedig beriker signaturen med en ny konstant $d : s_1 \diamond s_2$ og legger reglene $b \longrightarrow d$ og $c \longrightarrow d$ til $\mathcal{R}$ som da blir konfluent og typeminskende. Se [GKK90] for flere detaljer og fallgruber med denne fremgangsmåten.

Alternativt kunne man sikkert "senke" typen til c til $s_1 \diamond s_2$ og orientert likningen $b \longrightarrow c$. Det forskes endel rundt disse idéene i Frankrike i om dagen [HKK94].

Den interesserte kan her ha mye å jobbe videre med, men disse temaene går langt utenom skopet til denne oppgaven.

[6] Definisjon i draft-versjonen [CH93] sier ikke at u skal være av en mindre type enn $l\sigma$ og $r\sigma$ men dette *må* følge fra andre beviser i [CH93].

5.6 Sammendrag

Dette kapitlet har vært en kort innføring i typeordnet komplettering der jeg stort sett har villet indikere hovedforskjellene mellom utypet og typeordnet komplettering. Forkjellene og problemene ligger at Birkhoffs teorem ikke gjelder generelt, i typeminskningskravet, i typeordnet unifikasjon og kanskje særlig i variabelmengdene dersom man jobber med tomme typer. Kanskje er dette problemfelter andre interesserte vil se mer på?

Kapitlet har også motivert hvorfor nesten all forskning om typeordnet omskrivning synes å dreie seg om typeminskende systemer. Betraktninger omkring kritiske par gjør at selv ikke kompatibilitet kan erstatte typeminskningskravet. Til sist har vi fått vite hvorfor man så ofte må nevne variabelmengder eksplisitt i typeordnede sammenhenger.

Del II

Terminering av typeordnet omskrivning

Kapittel 6

Speiling – en måte å påvise typeordnet terminering

Typeordnet omskrivning har som nevnt vært gjenstand for stor interesse den senere tid[1]. Det kan derfor synes litt underlig at det, såvidt jeg kjenner til, bare er publisert én artikkel, Gnaedig [Gn92a], om terminering av typeordnet omskrivning.

Den enkleste måten å vise at et typeordnet omskrivningssystem terminerer er å se på systemet som et utypet system og vise at "den utypede varianten" terminerer, siden enhver typeordnet omskrivningssekvens $t_1 \longrightarrow t_2 \longrightarrow \cdots$ også finnes i det tilsvarende utypede omskrivningssystemet. Denne metoden er rask og brukes i dag bl. a. i implementasjonen av ELIOS-OBJ [Gn92b]. En av grunnene til dette er det går forholdsvis raskt, en annen grunn er sannsynligvis at det ikke finnes noen teori om feltet, da heller ikke teoretiske artikler om typeordnet omskrivning presenterer typeordnet terminering som noe annet enn terminering av det tilsvarende utypede systemet (se f. eks. [CH93] og [GKK90]).

Denne metoden å påvise typeordnet terminering, som jeg vil kalle *TT0*, er ikke spesielt sterk, da mange typeordnede omskrivningssystemer terminerer der den utypede varianten ikke ville terminere. Gnaedig [Gn92a] presenterer følgende eksempel på et slikt system:

Eksempel 6.1 Omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ avgjør hvorvidt et heltall er et partall ved først å gjøre om negative tall til positive.

Typer: Int, $Zero$, $NzNeg$, Neg, $NzPos$, Pos, $Bool$
$Zero \leq Neg \leq Int$
$Zero \leq Pos \leq Int$
$NzNeg \leq Neg$
$NzPos \leq Pos$

Funksjoner: $0 : Zero$
$s : Pos \rightarrow NzPos$
$p : Neg \rightarrow NzNeg$
$true : Bool$
$false : Bool$
$is_even : Int \rightarrow Bool$

[1] På instituttbiblioteket finnes f. eks. mer enn 50 artikler om typeordnet algebra og omskrivning.

$$
\begin{array}{ll}
 & is_even : NzPos \rightarrow Bool \\
 & is_even : NzNeg \rightarrow Bool \\
 & opp : NzNeg \rightarrow NzPos \\
\textbf{Regler:} & is_even(0) \longrightarrow true \\
 & is_even(s(0)) \longrightarrow false \\
 & (\forall x{:}Pos)\ is_even(s(s(x))) \longrightarrow is_even(x) \\
 & (\forall y{:}NzNeg)\ is_even(y) \longrightarrow is_even(opp(y)) \\
 & opp(p(0)) \longrightarrow s(0) \\
 & (\forall y{:}NzNeg)\ opp(p(y)) \longrightarrow s(opp(y))
\end{array}
$$

Den utypede varianten av systemet er ikke terminerende fordi det der ville ha vært en uendelig sekvens

$$is_even(y) \longrightarrow is_even(opp(y)) \longrightarrow is_even(opp(opp(y))) \longrightarrow \cdots$$

Det typeordnede systemet $\mathcal{R}$ terminerer imidlertid fordi termen $opp(y{:}NzNeg)$ er av typen $NzPos$ slik at den fjerde regelen *ikke* kan brukes for å omskrive $is_even(opp(y))$.

Merk også at deklarasjonene

$$
\begin{array}{l}
is_even : NzPos \rightarrow Bool \\
is_even : NzNeg \rightarrow Bool
\end{array}
$$

er fullstendig overflødige og bare brukes for at Gnaedig [Gn92a] skal kunne vise terminering av systemet. Fremgangsmåten jeg presenterer i denne oppgaven vil ikke være avhengig av slike unødvendige deklarasjoner.

Det fins flere måter å angripe problemet med å vise typeordnet terminering. En idé kunne være å lage helt nye typeordnede termineringsordninger. Det er antagelig ikke så lett å lage en slik ny ordning, da den også ville være en termineringsordning for utypet omskrivning (som er et spesialtilfelle av typeordnet omskrivning), noe som er omhandlet i utallige artikler. Det kan derfor se ut som om det er mer å hente på å tilpasse eksisterende ordninger til typeordnede systemer.

Gnaedig [Gn92a] "merker" funksjonssymbolene med tilhørende ariteter. I eksempel 6.1 kunne symbolet is_even merkes med argumentene $NzPos$ og $NzNeg$ slik at den fjerde regelen kan skrives

$$is_even_{NzNeg}(y) \longrightarrow is_even_{NzPos}(opp_{NzNeg}(y))$$

Vi ser på is_even_{NzNeg} og is_even_{NzPos} som to distinkte funksjonssymboler og kan da lett vise terminering av systemet med $\succ_{mpo}$ og presedensen $is_even_{NzNeg} \succ_F is_even_{NzPos} \succ_F opp_{NzNeg}$. For å bruke en slik metode, må Gnaedig i [Gn92a] først gjøre forandringer i både signaturen og regelmengden, og dette er bare mulig i en liten klasse såkalte *minimaliserbare* signaturer.

Min fremgangsmåte bygger også på en merking av funksjonssymboler, og jeg skylder derfor Isabelle Gnaedig en takk for denne idéen. Jeg bruker imidlertid disse *merkede funksjonssymbolene* til å lage noe jeg vil kalle en *speiling*, som er et utypet omskrivningssystem som skal representere det typeordnede systemet "så nøyaktig som mulig". Jeg tror ikke at det er mulig å lage en algoritme som finner en *perfekt speiling*, dvs. en speiling som terminerer hvis

og bare hvis det typeordnede systemet terminerer, uten å gå veien om Turing-maskiner. Jeg vil derfor nøye meg med å forlange at hver gang speilingen terminerer, så skal også det typeordnede systemet terminere. På denne måten kan jeg bruke utypede termineringsordninger for å vise terminering av speilingen, og dermed også av det opprinnelige systemet.

Hvordan speilingen skal se ut for at resultatet skal bli best mulig, avhenger også av hvilke ordninger man benytter. Jeg vil bruke stiordningene *mpo* og *lpo* både for å motivere fremgangsmåten og for å eksemplifisere teorien, som imidlertid skulle være forholdsvis generell.

Dette kapitlet omhandler den grunnleggende teorien om speilinger og fører frem til en metode *TT1*[2] for å påvise terminering av typeordnet omskrivning. I neste kapittel forbedres TT1 til en TT2-speiling som igjen forbedres til en TT3-speiling. Stiordningene viser seg være lite egnet til å vise terminering av TT3-speilingen. Knuth-Bendix-ordningen er klart bedre til dette formålet, men for å gi metoden full styrke moderniserer jeg denne til en (utypet) kvasiforenklingsordning $\succsim_{ukbo}$.

Det er to grunner til at jeg deler opp TT-metodene. For det første er TT3, som er en ren utvidelse av TT2, klart mer komplisert enn TT2, samtidig som TT2-metoden er ganske sterk. Det er derfor ofte tilstrekkelig å bruke TT2, og jeg vil derfor at TT2 skal være en fullverdig fremgangsmåte. Grunnen til det kraftige skillet mellom TT1 og TT2-metoden er mer bevisteknisk, slik at jeg slipper å gjenta beviser fra spranget fra TT1 til TT2 i korrekthetsbeviset for TT3-metoden.

Oppgaven fortsetter så med en oppsummering av tidligere arbeid innen typeordnet terminering, Gnaedigs [Gn92a], og i det nest siste kapitlet kommer en oppsummering og en vurdering av min fremgangsmåte. I det aller siste kapitlet prøver jeg å peke på hvor man kan finne nøklene til typeordnede termineringsordninger, både generelt og ut i fra metoden presentert i denne oppgaven. Jeg vil også si noen ord om hvilke områder jeg ikke tror man trenger å lete i.

Antagelser videre i oppgaven

Jeg vil ikke ha behov for monotoni eller regularitet av signaturer fremover. Det vil være tilstrekkelig å forlange at signaturene skal være preregulære, det vil si at enhver term skal ha en éntydig minste type.

En annen helt nødvendig antagelse i denne oppgaven er at alle reglene skal være typeminskende. Jeg har tidligere i oppgaven argumentert for at det er naturlig å forlange regularitet og typeminskning. Gnaedig forlanger i sin artikkel om typeordnet terminering [Gn92a] både regularitet og typeminskning (og minimaliserbarhet).

For å sikre beregnbarhet av min fremgangsmåte er jeg til sist nødt til å anta at enhver type i signaturen bare har et endelig antall typer som er mindre enn seg selv.

6.1 Terminering av typeordnede omskrivningssystemer

Mangelen på publiserte artikler om terminering av typeordnet omskrivning har ført til at begrepet terminering ikke engang er definert (som noe annet enn terminering av det tilsvar-

[2] TT står for det noe fantasiløse navnet "Typeordnet terminering" jeg har valgt å gi min metode.

ende utypede systemet [CH93]) for typeordnede omskrivningssystemer. Jeg definerer derfor begrepet i dette avsnittet som også kommer inn på noen andre egenheter ved typeordnet terminering.

I og med at utypet omskrivning kan ses på som et spesialtilfelle av typeordnet omskrivning kan de fleste resultatene fra utypet terminering overføres til typeordnet terminering. Det er da naturligvis også generelt uavgjørbart hvorvidt et typeordnet system terminerer.

6.1.1 Definisjon av terminering – tomme typer

Til forskjell fra utypede omskrivningssystemer gjør muligheten av tomme typer at det ikke engang er trivielt å definere terminering.

Eksempel (forts. av eks. 5.4) La $\{(\forall x : Tom)\ T \longrightarrow F\ ,\ (\forall x : Tom)\ F \longrightarrow T\}$ være regelmengden $\mathcal{R}$ i et omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$ der Σ er signaturen i eks. 5.4. Da fins det som vi har sett ingen termer t, t' slik at $t \xrightarrow[\mathcal{R}]{} {}_{\emptyset}\ t'$ er en omskrivning, og det fins følgelig ingen uendelige $\xrightarrow[\mathcal{R}]{} {}_{\emptyset}$-omskrivningssekvenser. Derimot finnes det en uendelig $\xrightarrow[\mathcal{R}]{} {}_{\{y:Tom\}}$-sekvens

$$T \xrightarrow[\mathcal{R}]{} {}_{\{y:Tom\}}\ F \xrightarrow[\mathcal{R}]{} {}_{\{y:Tom\}}\ T \xrightarrow[\mathcal{R}]{} {}_{\{y:Tom\}}\ \cdots$$

Er $\mathcal{R}$ et terminerende omskrivningssystem?

I lys av ovenstående eksempel velger jeg å definere terminering av typeordnede omskrivningssystemer på følgende måte:

Definisjon 6.1 (Terminering) *Et typeordnet omskrivningssystem* $\langle \Sigma, \mathcal{R} \rangle$ terminerer relativt til en variabelmengde X *dersom det ikke finnes en uendelig omskrivningssekvens*

$$t_1 \xrightarrow[\mathcal{R}]{} {}_X\ t_2 \xrightarrow[\mathcal{R}]{} {}_X\ t_3 \xrightarrow[\mathcal{R}]{} {}_X\ \cdots$$

av termer i $\mathcal{T}(\Sigma, X)$.

Jeg skriver at $\mathcal{R}$ *terminerer når* $\langle \Sigma, \mathcal{R} \rangle$ *terminerer relativt til en variabelmengde* X *som inneholder minst én variabel av hver type i signaturen. Jeg skriver videre at* $\mathcal{R}_Y$ *terminerer når* $\langle \Sigma, \mathcal{R} \rangle$ *terminerer relativt til variabelmengden* Y.

Denne oppgaven beskriver fremgangsmåter for å vise at et omskrivningssystem $\mathcal{R}$ terminerer relativt til en variabelmengde som inneholder alle typer. Ofte brukes omskrivningssystemer for å beregne uttrykk bestående av grunntermer (f. eks. $(SSS0 - S0)!$ og ikke $(x - y)!$) slik at det vel så interessant å vite hvorvidt $\mathcal{R}_\emptyset$ terminerer. Jeg ønsker derfor å kunne vise terminering av $\mathcal{R}_\emptyset$ ved å bruke teknikker for å vise terminering av $\mathcal{R}_X$.

For å vise at $\mathcal{R}$ terminerer relativt til $\emptyset$, kan man først fjerne alle reglene som ikke kan brukes i en $\xrightarrow[\mathcal{R}]{} {}_{\emptyset}$-omskrivning, dvs. alle regler $(\forall X)\ l \longrightarrow r$ der det fins en $x \in X_s$ samtidig som det ikke finnes noen grunnterm i $\mathcal{T}(\Sigma, \emptyset)_s$. I eks. 5.4 måtte man da fjerne begge reglene. Følgende eksempel viser at dette ikke er tilstrekkelig for å oppnå at $\mathcal{R}_X$ skal terminere hvis og bare hvis $\mathcal{R}_\emptyset$ terminerer:

Eksempel 6.2 La omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være gitt ved

Typer: s, s', s''
$s' \leq s$
$s'' \leq s$
Funksjoner: $f : s \rightarrow s$
$g : s' \rightarrow s$
$g : s'' \rightarrow s''$
$a : s'$
Regel: $(\forall x : s')\ f(x) \longrightarrow f(g(x))$

Det fins en grunnterm av type s' som gjør at regelen ikke kan fjernes. Det fins ingen uendelig $\underset{\mathcal{R}}{\longrightarrow}_{\emptyset}$-omskrivningssekvens, men derimot en uendelig $\underset{\mathcal{R}}{\longrightarrow}_{X}$-sekvens

$$f(y : s'') \underset{\mathcal{R}}{\longrightarrow}_X f(g(y)) \underset{\mathcal{R}}{\longrightarrow}_X f(g(g(y))) \underset{\mathcal{R}}{\longrightarrow}_X \cdots$$

når $y \in X_{s''}$.

Løsningen på dette problemet er å fjerne alle typer s og alle deklarasjoner som omtaler s fra signaturen Σ dersom mengden $\mathcal{T}(\Sigma, \emptyset)_s$ er tom. Dette kan gjøres veldig enkelt og vi har oppnådd at $\mathcal{R}_\emptyset$ terminerer hvis og bare hvis $\mathcal{R}_X$ terminerer. Fremgangsmåten i denne oppgaven kan derfor også benyttes til å vise at et omskrivningssystem terminerer for grunntermer.

Ellers vil jeg her bare nevne som en kuriositet at dersom et system ikke er typeminskende, eller mer nøyaktig ikke er kompatibelt, så kan det faktisk finnes gjentagende systemer som terminerer.

Eksempel 6.3 La den ikke-typeminskende omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være

Typer: s, s', s''
$s' \leq s$
$s'' \leq s'$
Funksjoner: $f : s' \rightarrow s$
$f : s'' \rightarrow s'$
$a : s''$
Regel: $(\forall x : s'')\ f(x) \longrightarrow f(f(x))$

Systemet $\mathcal{R}$ er gjentagende *og* terminerende siden termen $f(f(x))$ ikke kan skrives om, da $f(f(f(x)))$ ikke er en velformet term.

Et gjentagende og kompatibelt system er imidlertid alltid ikke-terminerende.

6.1.2 Retrakter og terminering

Når et system $\langle \Sigma, \mathcal{R} \rangle$ terminerer skulle man tro at også systemet utvidet med retraktdeklarasjoner og retraktregler på formen

$$\textbf{if } x \text{ er av typen } s' \textbf{ then } r_{s > s'}(x) \longrightarrow x$$

også terminerer. Følgende eksempel viser at det ikke forholder seg på denne måten:

Eksempel 6.4 La omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være

$$
\begin{array}{ll}
\textbf{Typer:} & s, s' \\
 & s' \leq s \\
\textbf{Funksjoner:} & f : s', s', s' \rightarrow s \\
 & a : s \\
 & b : s' \\
 & c : s' \\
\textbf{Regler:} & (\forall x : s')\ f(b, c, x) \longrightarrow f(x, x, x) \\
 & a \longrightarrow b \\
 & a \longrightarrow c
\end{array}
$$

$\mathcal{R}$ terminerer, da x ikke kan instansieres med a. Innfører vi en retrakt $r : s \rightarrow s'$ med tilhørende retraktregel har vi en uendelig omskrivningssekvens

$$
\begin{array}{l}
f(b, c, r(a)) \underset{\mathcal{R}}{\longrightarrow} f(r(a), r(a), r(a)) \overset{a \rightarrow b}{\underset{\mathcal{R}}{\longrightarrow}} f(r(b), r(a), r(a)) \overset{a \rightarrow c}{\underset{\mathcal{R}}{\longrightarrow}} \\
f(r(b), r(c), r(a)) \overset{r(x) \rightarrow x}{\underset{\mathcal{R}}{\longrightarrow}} f(b, r(c), r(a)) \overset{r(x) \rightarrow x}{\underset{\mathcal{R}}{\longrightarrow}} f(b, c, r(a)) \underset{\mathcal{R}}{\longrightarrow} \cdots
\end{array}
$$

For ikke å komplisere bildet mer enn nødvendig, vil jeg ta utgangspunkt i systemer uten retrakter i min metode og vil deretter vise hvordan metoden kan utvides til å vise terminering av systemer *med* retrakter.

Ved siden av ovennevnte tilfeller er det få teoretiske overraskelser i typeordnet terminering, ihvert fall så langt denne oppgaven strekker seg.

6.2 Speilinger

Jeg skal nå se på idéen med å merke funksjonssymboler med tilhørende aritet for så å definere en *speiling*, en utypet regelmengde som i termineringshenseende ligger så nært opptil det opprinnelige systemet som mulig.

Jeg ønsker for et gitt typeordnet omskrivningssystem $\mathcal{R}$ over en termalgebra $\mathcal{T}(\Sigma, X)$ finne en speiling R', et utypet omskrivningssystem over en utypet termmengde $T(F, V)$, slik at $\mathcal{R}$ terminerer hvis R' gjør det. På denne måten kan man fastslå terminering av $\mathcal{R}$ ved å fastslå terminering av R'.

En speiling av et typeordnet omskrivningssystem inn i et utypet ditto kan defineres på ulike måter. Ethvert typeordnet omskrivningssystem kan som kjent representeres av en Turing-maskin, som i sin tur kan modelleres av et utypet omskrivningssystem. Dette blir litt for komplisert samtidig som det ikke hjelper så mye da det generelt er uavgjørbart hvorvidt en Turing-maskin stopper. Jeg vil derfor bare se på en bestemt klasse speilinger, nemlig den klassen jeg vil kalle *syntaktiske speilinger*.

6.2.1 Syntaktisk speiling

I en syntaktisk speiling ser vi bare på typene til argumentene til en funksjon, og bygger opp en utypet term av en typeordnet term utifra en analyse av disse typene. Generelt vil jeg gjøre om en term t i den typeordnede algebraen $\mathcal{T}(\Sigma, X)$ til en term i en utypet

termalgebra $T(F,V)$ ved å foreta en merking av hvert funksjonssymbol i t. Den utypede funksjonssymbolmengden F blir da en mengde av merkede Σ-funksjonssymboler. En *funksjonssymbolsmerkingsfunksjon* finner ut hvordan et symbol i t skal merkes.

Definisjon 6.2 *En funksjonssymbolsmerkingsfunksjon* $\lfloor\ \rfloor_F : \Sigma \times S^* \rightarrow F$ *der* S *er mengden av typer i signaturen* Σ *er en funksjon som tar som argumenter et funksjonssymbol* f *fra* Σ *og en sekvens av typer* w *der* $|w| = |ar(f)|$ *og finner det tilhørende funsksjonssymbolet i* F.

Eksempel 6.5 La den typeordnede signaturen Σ være

Typer: s, s'
$s' \leq s$
Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s'$
$a : s$
$b : s'$

Mengden F av merkede funksjonssymboler kan være gitt $F = \{a, b, f_s, f_{s'}\}$. En funksjonssymbolsmerkingsfunksjon $\lfloor\ \rfloor_F$ kan defineres ved $\lfloor a \rfloor_F = a$, $\lfloor b \rfloor_F = b$, $\lfloor f, s \rfloor_F = f_s$ og $\lfloor f, s' \rfloor_F = f_{s'}$.

En *termspeilingsfunksjon* utfører merkingen av hele termen.

Definisjon 6.3 (Termspeilingsfunksjon) *En funksjonssymbolsmerkingsfunksjon* $\lfloor\ \rfloor_F : \Sigma \times S^* \rightarrow F$ *kan løftes til en* termspeilingsfunksjon $\lfloor\ \rfloor : \mathcal{T}(\Sigma, X) \rightarrow T(F,V)$ *ved at*

$$\lfloor f(t_1, \ldots, t_n) \rfloor = \lfloor f, LS(t_1), \ldots, LS(t_n) \rfloor_F (\lfloor t_1 \rfloor, \ldots, \lfloor t_n \rfloor)$$

Denne forholdsvis "løse" definisjonen kan være nyttig å ha for senere teoretiske betraktninger omkring styrken i speilinger. Jeg kommer imidlertid se på en mer begrenset klasse, nemlig den der variable speiles på variable i tillegg til det ovenstående.

Vi har tidligere sett at det er trivielt å beregne minste typen $LS(t)$ til en term t sålenge signaturen er preregulær. Funksjonen $\lfloor\ \rfloor$ er definert slik at lar en $\lfloor\ \rfloor_F$ merke hvert funksjonssymbol ut i fra kjennskap til minste typen til hvert argument. Kjenner man man minste typen så kjenner man jo også all annen typeinformasjon om parametrene. En kan også spørre seg hva ariteten til F-funksjonssymbolet $\lfloor f, LS(t_1), \ldots, LS(t_n) \rfloor_F$ skal være. Opplagt må den være større eller lik n. Man kunne tenke seg å laste på noen ekstra konstanter for å beholde mer typeinformasjon, slik at $f(x:s)$ kunne speiles av f. eks. $f_s(x, k_s)$, der k_s er en ny konstant. Dette er heller ikke nødvendig, da en slik informasjon også kan legges i "merkene" til f, dvs. den kan beregnes av $\lfloor\ \rfloor_F$.

Det er lett å se at termene t og $\lfloor t \rfloor$ alltid har samme "trestruktur" og at funksjonen $\lfloor\ \rfloor$ er en homomorfi fra $\mathcal{T}(\Sigma, X)$ til $T(F,V)$.

Eksempel (forts. av eks. 6.5) Med tidligere gitte F og $\lfloor\ \rfloor_F$ vil eksempelvis $\lfloor f(a) \rfloor = \lfloor f, LS(a) \rfloor_F (\lfloor a \rfloor) = f_s(a)$ og $\lfloor f(f(x:s)) \rfloor =$
$\lfloor f, LS(f(x:s)) \rfloor_F (\lfloor f(x:s) \rfloor) = f_s(\lfloor f, LS(x:s) \rfloor_F (\lfloor x:s \rfloor)) = f_s(f_s(x))$.

Jeg kan nå definere en syntaktisk speiling:

Definisjon 6.4 (Syntaktisk speiling) *En* syntaktisk speiling *av et trippel* $\langle \Sigma, X, \mathcal{R} \rangle$ *der* $\langle \Sigma, \mathcal{R} \rangle$ *er et typeordnet omskrivningssystem over* $\mathcal{T}(\Sigma, X)$ *er et kvadruppel* $\langle F, V, \lfloor\ \rfloor, R' \rangle$, *der* F *er en utypet signatur,* V *en utypet variabelmengde,* $\lfloor\ \rfloor$ *en termspeilingsfunksjon fra* $\mathcal{T}(\Sigma, X)$ *inn i den utypede termmengden* $T(F, V)$ *og* R' *et utypet omskrivningssystem over* $T(F, V)$ *slik at for alle termer* $t, t' \in \mathcal{T}(\Sigma, X)$ *så gjelder*

$$t \xrightarrow[\mathcal{R}]{} t' \Longrightarrow \lfloor t \rfloor \xrightarrow[R']{+} \lfloor t' \rfloor$$

Jeg nevner ikke variabelmengdene eksplisitt dersom de ikke skulle ha spesiell interesse.

Eksempel (forts. av eks. 6.5) La den typeordnede regelmengden $\mathcal{R}$ være $\{a \longrightarrow b\}$. En speiling $\langle F, \lfloor\ \rfloor, R' \rangle$ av $\langle \Sigma, \mathcal{R} \rangle$ kan være gitt ved en regelmengde $R' = \{a \longrightarrow b, f_s(x) \longrightarrow f_{s'}(x)\}$ i tillegg til F og $\lfloor\ \rfloor$ bestemt tidligere. Det er lett å se at $\langle F, \lfloor\ \rfloor, R' \rangle$ er en speiling av $\langle \Sigma, \mathcal{R} \rangle$ fordi $t \xrightarrow[\mathcal{R}]{} t' \Longrightarrow \lfloor t \rfloor \xrightarrow[R']{+} \lfloor t' \rfloor$ for alle termer t, $t' \in \mathcal{T}(\Sigma)$.

Dersom R' *ikke* inneholdt regelen $f_s(x) \longrightarrow f_{s'}(x)$ ville $\langle F, \lfloor\ \rfloor, R' \rangle$ ikke vært en speiling fordi $f(a) \xrightarrow[\mathcal{R}]{} f(b)$ er en omskrivning i $\mathcal{R}$ uten at vi ville hatt $\lfloor f(a) \rfloor = f_s(a) \xrightarrow[R']{+} f_{s'}(b) = \lfloor f(b) \rfloor$.

Det er ikke opplagt at en variabel må speiles inn i en variabel. Dersom man definerer $\lfloor x : s \rfloor = x$ mister man viktig typeinformasjon. For å ta vare typeinformasjonen kunne man vært fristet til å sette $\lfloor x : s \rfloor = k_s$ der k_s er en konstant. Dessverre kan ikke en variabel speiles inn i en konstant. Dette fordi vi kan ha en typeordnet omskrivning $f(x) \xrightarrow[\mathcal{R}]{} g(x)$ og som følge av dette av dette er det ofte uendelig mange omskrivninger $f(x\sigma) \xrightarrow[\mathcal{R}]{} g(x\sigma)$. Lar vi $\lfloor x : s \rfloor$ være en konstant, kan vi kanskje vise $\lfloor f(x) \rfloor \xrightarrow[R']{} \lfloor g(x) \rfloor$, men ikke $\lfloor f(x\sigma) \rfloor \xrightarrow[R']{+} \lfloor g(x\sigma) \rfloor$ for uendelig mange substitusjoner σ siden en omskrivningssekvens aldri kan introdusere en variabel. Vil man likevel beholde typeinformasjon og variable, kunne $\lfloor x : s \rfloor$ være lik en term $k_s(x)$. En slik fremgangsmåte kan være teoretisk interessant, men vil ikke hjelpe meg med de ordningene jeg kommer til å se på. Jeg velger derfor å definere $\lfloor x : s \rfloor = x$ i resten av oppgaven. Variabelmengden V i en speiling settes derfor lik den typeordnede variabelmengden X utvidet etter behov.

Når jeg i det videre skriver speiling, mener jeg syntaktisk speiling, og dersom $\langle F, V, \lfloor\ \rfloor, R' \rangle$ er en speiling av $\langle \Sigma, X, \mathcal{R} \rangle$ skriver jeg også at R' er en (regel-) speiling av $\mathcal{R}$.

Grunnen til at jeg bruker speilinger, er at man da kan bruke ordninger for utypede systemer for å vise terminering, noe som sammenfattes i følgende teorem:

Teorem 6.5 *Dersom* $\langle F, V, \lfloor\ \rfloor, R' \rangle$ *er en speiling av* $\langle \Sigma, X, \mathcal{R} \rangle$ *og* R' *terminerer, så terminerer* $\mathcal{R}$.

Bevis: Anta R' terminerer, men at $\mathcal{R}$ *ikke* gjør det. Da har vi en uendelig avledningssekvens $t_1 \xrightarrow[\mathcal{R}]{} t_2 \xrightarrow[\mathcal{R}]{} t_3 \xrightarrow[\mathcal{R}]{} \cdots$ i $\mathcal{R}$. Fordi $\langle F, V, \lfloor\ \rfloor, R' \rangle$ er en speiling av $\langle \Sigma, X, \mathcal{R} \rangle$ må vi per definisjon ha en uendelig sekvens $\lfloor t_1 \rfloor \xrightarrow[R']{+} \lfloor t_2 \rfloor \xrightarrow[R']{+} \lfloor t_3 \rfloor \xrightarrow[R']{+} \cdots$, noe som motsier antagelsen at R' skulle terminere. □

Foreløpig kan vi intuitivt si at mengden F består av funksjonssymbolene i Σ merket med tilhørende aritet. Disse "funksjonssymboler med aritet" regnes som forskjellige funksjonssymboler i den utypede R'.

Eksempel 6.6 La systemet $\langle \Sigma, \mathcal{R} \rangle$ være

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s$
	$g : s' \rightarrow s$
Regel:	$(\forall x : s')\ \ f(x) \longrightarrow f(g(x))$

En speiling $\langle F, \lfloor\ \rfloor, R' \rangle$ kunne gi oss $F = \{f_s, f_{s'}, g_{s'}\}$, $\lfloor f(x : s') \rfloor = f_{s'}(x)$, $\lfloor f(g(x)) \rfloor = f_s(g_{s'}(x))$ og $R' = \{f_{s'}(x) \longrightarrow f_s(g_{s'}(x))\}$. Med den opplagte utvidelsen av $\lfloor\ \rfloor_F$ og en V er dette en speiling av $\langle \Sigma, X, \mathcal{R} \rangle$. Den typede varianten terminerer, fordi regelen aldri kan brukes mer en én gang for hver forekomst av f, siden termen $g(x)$ er av typen s, som ikke er lik s' eller en subtybe av s'. Speilingen terminerer også, og det kan vises med f. eks. *mpo* med presedens $f_{s'} \succ_F f_s \succ_F g_{s'}$. Dette eksempelet kan leseren forøvrig merke seg med en gang, fordi det vil dukke opp flere ganger utover denne oppgaven. Sett nå $R'' = R' \cup \{f_s(x) \longrightarrow f_{s'}(x)\}$. Dette er også en speiling av $\mathcal{R}$, men R'' terminerer ikke!

Spørsmålet er om det i det generelle tilfellet er mulig å finne noe jeg velger å kalle en *perfekt syntaktisk speiling*, dvs. en speiling som terminerer hvis og bare hvis det opprinnelige systemet terminerer. Jeg har prøvd henholdsvis å motbevise og bevise eksistensen av en algoritme som gitt $\langle \Sigma, R \rangle$ gir oss en slik perfekt syntaktisk speiling. Jeg har imidlertid ikke kommet frem til noe konkret svar, men etter altfor mange forsøk antar jeg:

Gjetting 6.6 *Det fins ingen algoritme som finner en perfekt syntaktisk speiling gitt en generell* $\langle \Sigma, \mathcal{R} \rangle$.

Dette betyr at vi med utypede ordninger aldri kan vise terminering av så mange systemer som ønsket, noe som ikke er overraskende da terminering er uavgjørbart.

Denne oppgaven presenterer tre (litt forskjellige) måter å lage en speiling på. I resten av dette kapitlet presenteres det jeg vil kalle *TT1-speilingen* av et typeordnet omskrivningssystem.

6.3 Merking av funksjonssymbolene – beregning av $\lfloor\ \rfloor_F$

Det virket som en god idé å merke funksjonssymbolene med tilhørende ariteter, og se på et slikt merket symbol som et eget funksjonssymbol i speilingen. Spørsmålet er hvordan denne merkingen skal foregå. En idé kunne være å merke et symbol med verditypen til funksjonen.

Eksempel (forts. av eks. 6.1) Funksjonssymbolet *is_even* har verditype *Bool* og termen $is_even(y)$ kunne da merkes $is_even_{Bool}(y)$. Termen $is_even(opp(y))$ merkes da på samme måte $is_even_{Bool}(opp_{NzPos}(y))$. Med en slik merking gjelder ikke

$$is_even_{Bool}(y) \succ is_even_{Bool}(opp_{NzPos}(y))$$

for noen velfundert utypet termineringsordning $\succ$.

Det gir som vi ser ikke gode nok resultater å merke et funksjonssymbol med verditypen. Merker man funksjonssymbolene ut i fra (minste) typene til *argumentene* blir termene $is_even(y : NzNeg)$ og $is_even(opp(y : NzNeg))$ som vi tidligere har sett merket hhv. $is_even_{NzNeg}(y)$ og $is_even_{NzPos}(opp_{NzNeg}(y))$. Man kunne tenke seg å ville skvise enda mer informasjon i merkene ved å gi med *både* ariteten og verditypen, slik at $is_even(y)$ merkes $is_even_{NzNeg \to Bool}(y)$. Dette er helt unødvendig, da verditypen (*Bool*) er éntydig bestemt ut i fra typen til argumentene.

Funksjonssymbolene skal altså merkes med typene til argumentene den tar, men hvilke typer? Man skiller mest ved å merke symbolene med så liten type som mulig. Termen $y : NzNeg$ er også av typen *Int*, det samme er $opp(y : NzNeg)$, og $is_even_{Int}(y) \succ is_even_{Int}(opp_{NzNeg}(y))$ gjelder ikke for noen velfundert termineringsordning $\succ$. Men er det egentlig så bra at merkingen skiller mest mulig på F-funksjonssymboler? Eksempel 6.15 viser at det i fremgangsmåten TT1-*mpo* faktisk kan være uheldig med et slikt skille. I neste kapittel vil vi kunne ha ekvivalens mellom F-funksjonssymboler, og da er det alltid best å skille mest mulig.

Ariteten et Σ-funksjonssymbol skal merkes med bør altså være "så lav som mulig", men nøyaktig hva den skal den være? Det er i prinsippet minst to fremgangsmåter for å definere mengden F av merkede funksjonssymboler.

6.3.1 Funksjonssymbolmengden F og funksjonen $\lfloor\ \rfloor_F$ – variant 1

Man kan definere mengden F av merkede funksjonssymboler

$$F = \{f_w \mid f \in \Sigma_{w \to s} \text{ for én eller annen } s\}$$

det vil si at hver deklarasjon i Σ gir opphav til ett funksjonssymbol i F. Ariteten $ar(f_w)$ til et symbol f_w er alltid lik lengden av sekvensen w.

Eksempel (forts. av eks. 6.1) Gitt signaturen Σ i eksempel 6.1 ville den tilhørende utypede funksjonssymbolmengden F vært

$$F = \{0, s_{Pos}, p_{Neg}, true, false, is_even_{Int}, is_even_{NzPos}, is_even_{NzNeg}, opp_{NzNeg}\}$$

Funksjonssymbolsmerkingsfunksjonen $\lfloor\ \rfloor_F$ kan da defineres slik:

$\lfloor f, s_1, \ldots, s_n \rfloor_F = f_{s'_1,\ldots,s'_n}$ der $s'_1, \ldots, s'_n$ er den minste f-ariteten slik at $s_i \leq s'_i$ for alle i.

Følgende eksempel viser at signaturen må være regulær, eller ha tilsvarende egenskaper, for at $\lfloor\ \rfloor_F$ definert på denne måten skal være veldefinert.

Eksempel 6.7 La en signatur være gitt ved

Typer: $Int, Nat, Neg, Zero$
$Nat \leq Int$
$Neg \leq Int$
$Zero \leq Nat$
$Zero \leq Neg$
Funksjoner: $0 : Zero$
$f : Nat \rightarrow Int$
$f : Neg \rightarrow Int$
$\vdots$

Signaturen er ikke regulær, og $\lfloor f, Zero \rfloor_F$ er ikke veldefinert fordi den kan enten ha verdien f_{Nat} eller f_{Neg}. Hadde vi lagt til en deklarasjon $f : Zero \rightarrow Int$ ville signaturen vært regulær og funksjonene $\lfloor\ \rfloor_F$ og $\lfloor\ \rfloor$ veldefinerte.

Funksjonen $\lfloor\ \rfloor_F$ (og dermed også $\lfloor\ \rfloor$) er veldefinert og beregnbar når signaturen er regulær. Å velge $F = \{f_w | f \in \Sigma_{w \rightarrow s}\}$ har også en annen svakhet. Anta nemlig at de unødvendige deklarasjonene $is_even : NzNeg \rightarrow Bool$ og $is_even : NzPos \rightarrow Bool$ *ikke* er med i signaturen Σ. Den fjerde regelen i eks. 6.1 kan likefullt være

$$(\forall y : NzNeg)\ is_even(y) \longrightarrow is_even(opp(y))$$

Siden de ovennevnte deklarasjonene har forsvunnet er imidlertid

$$\lfloor is_even(y{:}NzNeg) \rfloor = \lfloor is_even, NzNeg \rfloor_F(y) = is_even_{Int}(y)$$

og

$$\lfloor is_even(opp(y{:}\ NzNeg)) \rfloor = \ldots = \lfloor is_even, NzPos \rfloor_F(\lfloor opp, NzNeg \rfloor_F(y)) = is_even_{Int}(opp_{NzNeg}(y))$$

og vi ender opp med å måtte vise

$$is_even_{Int}(y) \succ is_even_{Int}(opp_{NzNeg}(y))$$

som aldri gjelder for noen fornuftig ordning $\succ$.

6.3.2 En annen måte å definere F på

Den andre tilnærmingsmåten bøter på dette problemet ved å definere den utypede funksjonssymbolmengden

$$F = \{f_{w'} \mid \exists s | w' \leq w \wedge f \in \Sigma_{w \rightarrow s}\}$$

dvs. dersom $f : w \rightarrow s$ er en deklarasjon i Σ, så er $f_{w'}$ med i F for alle sekvenser av typer w' der $w' \leq w$. Denne definisjonen har en rekke fordeler. For det første er den finere oppdelt enn den forrige. Uansett om deklarasjonene $is_even : Nz \rightarrow Bool$ og $is_even : NzNat \rightarrow Bool$ er med i signaturen eller ikke, vil

$$F = \{\ldots, is_even_{Int}, is_even_{Nat}, is_even_{Neg}, is_even_{Zero}, is_even_{NzNeg}, is_even_{NzPos}, \ldots\}$$

fordi $is_even \in \Sigma_{Int \to Bool}$. Den andre fordelen med denne metoden er at det går meget raskt å beregne $\lfloor\ \rfloor_F$ fordi $\lfloor f, s_1, \ldots, s_n \rfloor_F$ etter denne siste definisjonene av $\lfloor\ \rfloor_F$ settes lik $f_{s_1,\ldots,s_n}$. Den tredje fordelen med å velge denne definisjonen av F er at det i dette tilfelle er tilstrekkelig å forlange at signaturen Σ er preregulær. Funksjonen $\lfloor\ \rfloor$ er i dette tilfellet alltid veldefinert, da $\lfloor f(t_1, \ldots, t_n) \rfloor = \lfloor f, LS(t_1), \ldots, LS(t_n) \rfloor_F(\ldots) = f_{LS(t_1),\ldots,LS(t_n)}(\ldots)$. Ordningene indusert av at F er definert på denne måten vil, som skal se senere, være minst like sterke som den forrige varianten. Ulempen med en slik større F er at man får mange fler symboler å jobbe med. Jeg ville uansett helt klart anbefale denne siste definisjonen av F, men overlater til brukeren å avgjøre dette ved å arbeide med begrepene F og $\lfloor\ \rfloor_F$ i denne oppgaven.

Man kan selvfølgelig kombinere variantene ved å utvide signaturen. Det eneste kravet man må stille er at den utvidede signaturen blir *kvasiregulær*, der en kvasiregulær signatur er en signatur som ville vært regulær dersom enhver for slapp deklarasjon $f : w \to s$ ble byttet ut med en deklarasjon $f : w \to LS(f(\overline{x} : w))$. Gitt en preregulær signatur kan denne faktisk alltid utvides til en kvasiregulær signatur, som i sin tur kan strammes inn til en regulær signatur. Jeg vil i mine eksempler bruke den første definisjonen av F, nemlig $F = \{f_w \mid f \in \Sigma_{w \to s}\}$, men dette er *utelukkende* for å øke leseligheten.

6.4 Funksjonssymbolsprang

Vi begynner dermed å bli klare for å kikke på regelmengden i en speiling. Omskrivningssystemet i eks. 6.1 var typebevarende, dvs. venstre- og høyresiden i hver regel hadde samme minste type for alle substitusjoner av variablene i regelen. Man er ikke alltid *så* heldig med regelmengdene. Eksempelvis ville den første likningen i $NatStack$-spesifikasjonen i eks. 3.1 orienteres til regel $(\forall s : Stack,\ n : Nat)\ pop(push(s, n)) \longrightarrow s$ hvis instans når $s = empty$ er ekte typeminskende. La oss derfor heller se på følgende system:

Eksempel 6.8 La det typeminskende systemet $\mathcal{R}$ være

Typer:	s, s_1, s_2
	$s_1 \leq s$
	$s_2 \leq s$
Funksjoner:	$f : s \to s$
	$f : s_1 \to s$
	$f : s_2 \to s$
	$g : s \to s$
	$g : s_1 \to s_2$
	$a : s$
	$b : s_1$
	$\vdots$
Regler:	$f(b) \longrightarrow f(g(b))$
	$a \longrightarrow b$

Termene kan speiles på måten beskrevet i forrige avsnitt slik at $\lfloor f(b) \rfloor = f_{s_1}(b)$ og $\lfloor f(g(b)) \rfloor = f_{s_2}(g_{s_1}(b))$. Systemet kan representeres i form av de utypede reglene $f_{s_1}(b) \longrightarrow f_{s_2}(g_{s_1}(b))$ og $a \longrightarrow b$. I en speiling $\langle F, \lfloor\ \rfloor, R' \rangle$ må enhver omskrivning

$t \xrightarrow[\mathcal{R}]{} t'$ kunne representeres av en omskrivningssekvens $\lfloor t \rfloor \xrightarrow[R']{+} \lfloor t' \rfloor$. Dette introduserer hovedproblemet med speilinger, nemlig vanskeligheten av å få dem lukket under regelapplikasjon. Med bruk av den andre regelen fins det en typeordnet omskrivning $f(g(a)) \xrightarrow[\mathcal{R}]{} f(g(b))$, slik at en speiling må oppfylle $\lfloor f(g(a)) \rfloor \xrightarrow[R']{+} \lfloor f(g(b)) \rfloor$. Siden $\lfloor f(g(a)) \rfloor = f_s(g_s(a))$ og $\lfloor f(g(b)) \rfloor = f_{s_2}(g_{s_1}(b))$ blir *alle* F-funksjonssymbolene forandret selv om den typeordnede omskrivningen foregår i posisjon 1.1! Det er lett å se at $R' = \{f_{s_1}(b) \longrightarrow f_{s_2}(g_{s_1}(b)),\ a \longrightarrow b\}$ *ikke* oppfyller $f_s(g_s(a)) \xrightarrow[R']{+} f_{s_2}(g_{s_1}(b))$ og derfor ikke er en speiling.

Bruk av en typeminskende (instans av en) regel kan altså gi opphav til at funksjonssymboler i $\lfloor t \rfloor$ forandres i $\lfloor t' \rfloor$ langt fra posisjonen der en typeomskrivning omskrivning $t \xrightarrow[\mathcal{R}]{} t'$ foregår. Jeg vil kalle disse "sideeffektene" av en typeordnet omskrivning for *F-funksjonssymbolsprang.*

Definisjon 6.7 (F-funksjonssymbolsprang) *Anta at det er gitt et typeordnet omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$, en utypet funksjonssymbolmengde F og en termspeilingsfunksjon $\lfloor\ \rfloor : \mathcal{T}(\Sigma, X) \rightarrow T(F, V)$. Anta videre at en typeordnet omskrivning $t \xrightarrow[\mathcal{R}]{} t'$ skjer i posisjon p i t. Det har skjedd et* F-funksjonssymbolsprang *i posisjon q dersom (F-)funksjonssymbolet i posisjon q i termen $\lfloor t \rfloor$ er forskjellig fra funksjonssymbolet i posisjon q i $\lfloor t' \rfloor$ og det* ikke *fins en (tom eller ikke-tom) posisjon r slik at $q = p.r$.*

Jeg skriver ofte *F-sprang* for F-funksjonssymbolsprang.

Eksempel (forts. av eks. 6.8) I omskrivningen $f(g(a)) \xrightarrow[\mathcal{R}]{} f(g(b))$ nevnt i eks. 6.8 er det to F-sprang, i posisjonene ϵ og 1.

Det viser seg som vi skal se at F-sprang alltid skjer i posisjoner i en term t mellom roten (posisjon ϵ) og posisjonen p der omskrivningen foretas, og jeg vil derfor få bruk for følgende definisjoner som også vil brukes i beviser i kapittel 8.

Definisjon 6.8 *For en term t og en posisjon $p = p_1.p_2.\ldots.p_n$ i t, defineres posisjonen $p(i)$, der $0 \leq i \leq n$, ved $p(i) = p_1.p_2.\ldots.p_{n-i}$.*

Posisjonen $p(i)$ angir altså posisjonen man får når man går i skritt fra posisjon p i t oppover den éntydige veien mot roten.

Eksempel 6.9 Anta at en posisjon $p = 1.2.1.3.2$. Da er $p(0) = 1.2.1.3.2$, $p(1) = 1.2.1.3$, $p(3) = 1.2$, $p(5) = \epsilon$ mens $p(8)$ er udefinert.

Definisjon 6.9 *Dybden D_p til en posisjon p er lik lengden til sekvensen p.*

I følgende lemmata antar jeg hele tiden at $t \xrightarrow[\mathcal{R}]{} t'$ er en typeordnet omskrivning med en regelapplikasjon i posisjon p og at $\lfloor\ \rfloor$ er utvidelsen av en funksjon $\lfloor\ \rfloor_F$ definert som beskrevet i avsnitt 6.3.

Lemma 6.10 *Det kan bare være F-sprang i posisjonene $p(0), p(1), \ldots, p(D_p)$.*

Bevis: Følger av at $\lfloor\ \rfloor$ er en homomorfi og at en term har trestruktur. □

Et F-sprang kan bare oppstå i en omskrivning dersom regelinstansen som er brukt til omskrivningen er ekte typeminskende, noe som fører til et F-sprang i posisjon $p(1)$. Men termen $t|_{p(1)}$ kan på grunn av monotoni-egenskaper til signaturen være av en større type enn $t'|_{p(1)}$, noe kan føre til et F-sprang i posisjon $p(2)$ osv., så for å gjøre noe med F-sprangene må man vite hvordan de ser ut. Siden jeg i denne oppgaven ikke har antatt at signaturene er monotone, trenger jeg følgende egenskap:

Lemma 6.11 *En preregulær signatur er LS-*monoton*, dvs.*

$$LS(t) \geq LS(t') \Longrightarrow LS(f(\ldots,t,\ldots)) \geq LS(f(\ldots,t',\ldots))$$

Bevis: Husk at $LS(f(t_1,\ldots,t_n)) = \min\{s \mid f \in \Sigma_{s_1,\ldots,s_n \to s} \wedge \forall i \in 1..n | LS(t_i) \leq s_i\}$. Anta så at $LS(f(\ldots,t,\ldots)) = s$ og $LS(f(\ldots,t',\ldots)) = s'$. Definisjonen av LS gir at s' ikke kan være større enn s. Situasjonen $s \bowtie s'$ kan heller ikke oppstå, fordi både s' og én eller annen $s'' \leq s$ ville vært minimale typer til $f(\ldots,t',\ldots)$, men det er umulig da preregularitet forlanger at enhver term har éntydig minste type. □

Lemma 6.12 *Anta at $s'_i \leq s_i$, $\lfloor f, s_1,\ldots,s_i,\ldots,s_n\rfloor_F = f_w$ og at $\lfloor f, s_1,\ldots,s'_i,\ldots,s_n\rfloor_F = f_{w'}$. Da er $w' \leq w$.*

Bevis: Uansett hvilken av de to tidligere nevnte definisjonene av F og $\lfloor\ \rfloor_F$ som brukes, følger $w' \leq w$ av definisjonen av $\lfloor\ \rfloor_F$ og det faktum at $\lfloor\ \rfloor_F$ er veldefinert. □

Disse lemmata fører til at vi kan karakterisere F-sprang der jeg skriver at termen t er lik et symbol f i posisjon p dersom $t|_p = f(t_1,\ldots,t_n)$ for $n \geq 0$.

Teorem 6.13 *La $t \xrightarrow[\mathcal{R}]{} t'$ være en typeordnet omskrivning i posisjon p. Dersom det forekommer et F-sprang i posisjon $p(i)$ $(1 \leq i \leq D_p)$ og $\lfloor t\rfloor$ er lik f_w i posisjon $p(i)$, og $\lfloor t'\rfloor$ er lik $g_{w'}$ på samme sted, så er $f = g$ og $w' < w$.*

Bevis: Definisjonen av $\lfloor\ \rfloor$ og $\lfloor\ \rfloor_F$ gir trivielt $f = g$. Siden omskrivningssystemet er typeminskende, gjelder $LS(t|_p) \geq LS(t'|_p)$. Itereres lemma 6.11 gir dette $LS(t|_{p(i-1)}) \geq LS(t'|_{p(i-1)})$. Termen $\lfloor t\rfloor$ er lik symbolet $\lfloor f, LS(t_1),\ldots,LS(t|_{p(i-1)}),\ldots,LS(t_n)\rfloor_F$ i posisjon $p(i)$. Tilsvarende er $\lfloor t'\rfloor$ lik $\lfloor f, LS(t_1),\ldots,LS(t'|_{p(i-1)}),\ldots,LS(t_n)\rfloor_F$ i posisjon $p(i)$. Lemma 6.12 sier da at $w' \leq w$, og siden det har forekommet et F-sprang, kan disse ikke være like, slik at $w' < w$. □

Teorem 6.13 indikerer at vi kan kompensere for F-sprang ved å innføre en regel

$$f_w(x_1,\ldots,x_n) \longrightarrow f_{w'}(x_1,\ldots,x_n)$$

(der variablene $x_1,\ldots,x_n$ er innbyrdes disjunkte) i speilingen for hvert par $f_w, f_{w'}$ i F der $w' < w$. Jeg velger å kalle en slik regel for en *ekstraregel* i R'.

Eksempel (forts. av eks. 6.8) Mengden av ekstraregler i systemet R' i eks. 6.8 blir da

$$\begin{array}{l} f_s(x) \longrightarrow f_{s_1}(x) \\ f_s(x) \longrightarrow f_{s_2}(x) \\ g_s(x) \longrightarrow g_{s_1}(x) \end{array}$$

Omskrivningen $f(g(a)) \xrightarrow[\mathcal{R}]{} f(g(b))$ kan da representeres i speilingen av sekvensen

$$\lfloor f(g(a)) \rfloor = f_s(g_s(a)) \xrightarrow[R']{a \to b} f_s(g_s(b)) \xrightarrow[R']{eks} f_{s_2}(g_s(b)) \xrightarrow[R']{eks} f_{s_2}(g_{s_1}(b)) = \lfloor f(g(b)) \rfloor$$

der $\xrightarrow[R']{eks}$ står for bruk av en ekstraregel i R'.

6.5 Regelmengden R' i TT1-speilingen

Den typeordnede regelmengden $\mathcal{R}$ skal nå speiles inn i den utypede regelmengden R'. Alle reglene $l \longrightarrow r$ i $\mathcal{R}$ må naturligvis representeres i R', slik at

$$\lfloor l \rfloor \longrightarrow \lfloor r \rfloor$$

er en regel i R'.

Siden en variabel $x:s$ i $l \longrightarrow r$ også kan instansieres med termer av lavere (minste) type enn s, viser følgende eksempel at det ikke er tilstrekkelig å bare ta med regelen $\lfloor l \rfloor \longrightarrow \lfloor r \rfloor$ og ekstrareglene i R'.

Eksempel 6.10 La det typeordnede omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være[3]

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \to s$
	$f : s' \to s$
	$b : s'$
Regel:	$(\forall x:s)\ \ f(x) \longrightarrow b$

Regelen speiles i $f_s(x) \longrightarrow b$ og vi har en ekstraregel $f_s(x) \longrightarrow f_{s'}(x)$. Termen $f(b)$ er en instans av $f(x:s)$, slik at $f(b) \xrightarrow[\mathcal{R}]{} b$ er en omskrivning i $\mathcal{R}$. Regelmengden $R' = \{f_s(x) \longrightarrow b,\ \ f_s(x) \longrightarrow f_{s'}(x)\}$ danner imidlertid ingen speiling av $\mathcal{R}$ fordi $\lfloor f(b) \rfloor = f_{s'}(b)$, og ingen regel i R' kan anvendes på denne termen. Vi har derfor ikke ønskede $\lfloor f(b) \rfloor \xrightarrow[R']{+} \lfloor b \rfloor$.

For å bøte på dette problemet tar jeg med regelen

$$\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$$

i (TT1-)regelspeilingen R' for alle spesialiseringer μ av variable i l for alle regler $l \longrightarrow r$ i $\mathcal{R}$. Jeg sier at instansen $l\mu \longrightarrow r\mu$ er en *regelspesialisering*.

[3]Deklarasjonen $f : s' \to s$ er unødvendig. Jeg tar den med for å indikere hvilken mengde F av merkede symboler jeg ønsker å jobbe med (i dette tilfellet f_s og $f_{s'}$).

Gitt en typeminskende typeordnet regelmengde $\mathcal{R}$, en (preregulær) typeordnet signatur Σ og en mengde F av utypede funksjonssymboler defineres regelmengden R' i en TT1-speiling av $\langle \Sigma, \mathcal{R} \rangle$ til å bestå av en mengde

$$\{\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor \mid l \longrightarrow r \in \mathcal{R} \wedge \mu \text{ spesialisering av variable i } l\}$$

som jeg kaller *opprinnelige regler* og en mengde

$$\{f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n) \mid f_w, f_{w'} \in F \wedge w' < w\}$$

av ekstraregler.

For å vise at $\langle F, \lfloor\ \rfloor, R' \rangle$ er en speiling trenger jeg følgende lemma:

Lemma 6.14 *For enhver term $t \in \mathcal{T}(\Sigma, X)$ og typeordnet substitusjon $\sigma : X \rightarrow \mathcal{T}(\Sigma, X)$ gjelder*

$$\lfloor t\sigma \rfloor = \lfloor t\mu \rfloor \sigma'$$

når μ er en spesialisering av X definert ved

$$\mu = \{x_1 \mapsto x_1 : LS(x_1\sigma), \ldots, x_n \mapsto x_n : LS(x_n\sigma)\}$$

og σ' er en utypet *substitusjon $\sigma' : V \rightarrow T(F, V)$ definert ved*

$$\sigma' = \{x_1 \mapsto \lfloor x_1\sigma \rfloor, \ldots, x_n \mapsto \lfloor x_n\sigma \rfloor\}$$

Bevis: Siden hver term har éntydig minste type og σ er en typeordnet substitusjon er μ og σ' veldefinerte. Jeg viser først hjelpesetningen $LS(u\sigma) = LS(u\mu)$ for enhver subterm u til t. Setningen vises ved induksjon på oppbygningen av u.

Tilfelle $u = x_i$**:** $LS(x_i\sigma) = LS(x_i\mu)$ gjelder per definisjon av μ.

Tilfelle $u = f(u_1, \ldots, u_m)$ **der** $m \geq 0$**:** Induksjonshypotesene sier $LS(u_i\sigma) = LS(u_i\mu)$ for alle i, slik at $u_i\sigma \in \mathcal{T}(\Sigma, X)_{s_i} \Longleftrightarrow u_i\mu \in \mathcal{T}(\Sigma, X)_{s_i}$ for alle typer s_i. Resultatet

$$LS(f(u_1\sigma, \ldots, u_m\sigma)) = LS(f(u_1\mu, \ldots, u_m\mu))$$

følger fra disse og definisjonen av LS-operatoren.

Gitt $LS(u\sigma) = LS(u\mu)$ kan setningen $\lfloor t\mu \rfloor \sigma' = \lfloor t\sigma \rfloor$ nå vises ved induksjon på oppbygningen av t.

Tilfelle $t = x_i$**:** $\lfloor x_i\mu \rfloor \sigma' = x_i\sigma' = \lfloor x_i\sigma \rfloor$.

Tilfelle $t = f(t_1, \ldots, t_m)$**:** Høyresiden $\lfloor f(t_1, \ldots, t_m)\sigma \rfloor$ er lik $\lfloor f(t_1\sigma, \ldots, t_m\sigma) \rfloor$ som ved definisjon av $\lfloor\ \rfloor$-funksjonen er lik

$$\lfloor f, LS(t_1\sigma), \ldots, LS(t_m\sigma) \rfloor_F (\lfloor t_1\sigma \rfloor, \ldots, \lfloor t_m\sigma \rfloor)$$

Venstresiden $\lfloor t\mu \rfloor \sigma'$ er ved definisjonen av $\lfloor\ \rfloor$-operatoren lik

$$(\lfloor f, LS(t_1\mu), \ldots, LS(t_m\mu) \rfloor_F (\lfloor t_1\mu \rfloor, \ldots, \lfloor t_m\mu \rfloor))\sigma'$$

Siden σ' er en homomorfi er uttrykket lik

$$\lfloor f, LS(t_1\mu), \ldots, LS(t_m\mu) \rfloor_F (\lfloor t_1\mu \rfloor \sigma', \ldots, \lfloor t_m\mu \rfloor \sigma')$$

Induksjonshypotesene sier nå $\lfloor t_i\sigma \rfloor = \lfloor t_i\mu \rfloor \sigma'$ slik at ovenstående uttrykk er lik

$$\lfloor f, LS(t_1\mu), \ldots, LS(t_m\mu) \rfloor_F (\lfloor t_1\sigma \rfloor, \ldots, \lfloor t_m\sigma \rfloor)$$

Da vi nettopp har vist $LS(t_i\sigma) = LS(t_i\mu)$ er dette uttrykket lik den ønskede høyresiden

$$\lfloor f, LS(t_1\sigma), \ldots, LS(t_m\sigma) \rfloor_F (\lfloor t_1\sigma \rfloor, \ldots, \lfloor t_m\sigma \rfloor)$$

□

Følgende teorem, oppgavens viktigste, sier at $\langle F, \lfloor\ \rfloor, R' \rangle$ er en speiling av $\langle \Sigma, \mathcal{R} \rangle$ uansett hvilken av de to tidligere nevnte fremgangsmåtene vi bruker for å definere F.

Teorem 6.15 *La $\langle \Sigma, \mathcal{R} \rangle$ være et typeminskende omskrivningssystem der Σ er en preregulær signatur. Da er $\langle F, \lfloor\ \rfloor, V, R' \rangle$ definert tidligere i dette kapitlet en speiling av $\langle \Sigma, X, \mathcal{R} \rangle$, dvs. for alle termer t, t' i $\mathcal{T}(\Sigma, X)$ gjelder*

$$t \xrightarrow[\mathcal{R}]{} t' \Longrightarrow \lfloor t \rfloor \xrightarrow[R']{+} \lfloor t' \rfloor$$

Bevis: Anta $t \xrightarrow[\mathcal{R}]{} t'$, dvs. det fins en regel $l \longrightarrow r$ i $\mathcal{R}$, en typeordnet substitusjon σ og en posisjon p i t slik at $t|_p = l\sigma$ og $t' = t[r\sigma]_p$. Vi lar så en spesialisering μ være gitt som i lemma 6.14 ved

$$\mu = \{x_1 \mapsto x_1 : LS(x_1\sigma), \ldots, x_n \mapsto x : LS(x_n\sigma)\}$$

Definisjonen av R' sier at R' inneholder regelen

$$\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$$

Vi lar også den *utypede* substitusjonen σ' fra $T(F, V)$ til $T(F, V)$ være definert som i lemma 6.14 ved

$$\sigma' = \{x_1 \mapsto \lfloor x_1\sigma \rfloor, \ldots, x_n \mapsto \lfloor x_n\sigma \rfloor\}$$

der $x_1, \ldots, x_n$ er variablene i $\lfloor l\mu \rfloor$ som "svarer til" variablene $x_1, \ldots, x_n$ i l.

Ved lemma 6.14 gjelder både $\lfloor l\sigma \rfloor = \lfloor l\mu \rfloor \sigma'$ og $\lfloor r\sigma \rfloor = \lfloor r\mu \rfloor \sigma'$. Da har vi $\lfloor t|_p \rfloor = \lfloor t \rfloor|_p = \lfloor l\sigma \rfloor = \lfloor l\mu \rfloor \sigma'$. Siden $\lfloor\ \rfloor$ er en homomorfi gjelder $\lfloor t \rfloor = \lfloor t \rfloor [\lfloor t \rfloor|_p]_p$. Bruk av regelen $\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$ i posisjon p med substitusjon σ' gir da en omskrivning

$$\lfloor t \rfloor = \lfloor t \rfloor [\lfloor t \rfloor|_p]_p = \lfloor t \rfloor [\lfloor l\mu \rfloor \sigma']_p \xrightarrow[R']{} \lfloor t \rfloor [\lfloor r\mu \rfloor \sigma']_p = \lfloor t \rfloor [\lfloor r\sigma \rfloor]_p$$

Problemet nå er at $\lfloor t \rfloor [\lfloor r\sigma \rfloor]_p$ ikke nødvendigvis er lik $\lfloor t[r\sigma]_p \rfloor$ fordi det kan ha forekommet F-sprang. Teorem 6.13 og oppbygningen av $\lfloor\ \rfloor$ gir da at $\lfloor t[r\sigma]_p \rfloor = \lfloor t' \rfloor$ og $\lfloor t \rfloor [\lfloor r\sigma \rfloor]_p$ bare skiller seg i posisjoner $p(i)$ $(1 \leq i \leq D_p)$ ved at et symbol f_s i disse posisjonene eventuelt skal erstattes med en $f_{s'}$ der $s' < s$. For alle slike $f_s, f_{s'}$ har vi en ekstraregel

$$f_s(x_1, \ldots, x_k) \longrightarrow f_{s'}(x_1, \ldots, x_k)$$

i R' som kan brukes i posisjon $p(i)$ for å foreta denne utskiftningen helt til man ender opp med $\lfloor t[r\sigma]_p \rfloor = \lfloor t' \rfloor$.

Omskrivningssekvensen blir altså

$$\lfloor t \rfloor = \lfloor t \rfloor [\lfloor l\mu \rfloor \sigma']_p \xrightarrow[R']{\lfloor l\mu \rfloor \to \lfloor r\mu \rfloor} \lfloor t \rfloor [\lfloor r\mu \rfloor \sigma']_p = \lfloor t \rfloor [\lfloor r\sigma \rfloor]_p \xrightarrow[R']{eks\,*} \lfloor t[r\sigma]_p \rfloor = \lfloor t' \rfloor$$

der $\xrightarrow[R']{eks}$ står for bruk av en ekstraregel i R'. Denne omskrivningssekvensen er lik den ønskede

$$\lfloor t \rfloor \xrightarrow[R']{+} \lfloor t' \rfloor$$

□

6.6 Termineringsprosedyrer indusert av TT1-speilingen

Speilinger gjør at man kan vise terminering av et typeordnet system $\mathcal{R}$ ved å vise terminering av den utypede speilingen R'.

Jeg velger å bruke *multiset stiordning* for å eksemplifisere fremgangsmåten. Det første man legger merke til er at alle ekstrareglene $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ er $\succ_{mpo}$ -minskende hvis og bare hvis presedensen $\succsim_F$ mellom F-funksjonssymbolene oppfyller $f_w \succ_F f_{w'}$ når $w' < w$. Presedensen $\succsim_F$ må derfor defineres slik at dette kravet er oppfylt.

Fremgangsmåten TT1-*mpo* for å vise terminering av et typeordnet system $\langle \Sigma, \mathcal{R} \rangle$ der Σ er preregulær og ingen type i Σ har uendelig mange typer under seg, og der $\mathcal{R}$ er en typeminskende regelmengde blir altså som vist i figur 6.1.

1. Definér den utypede funksjonssymbolmengden F og funksjonen $\lfloor\ \rfloor_F$ på én av måtene beskrevet i avsnitt 6.3.
2. La $\succsim_F$ være en kvasiordning over F som tilfredsstiller
$$f_w, f_{w'} \in F \wedge w' < w \Longrightarrow f_w \succ_F f_{w'}$$
3. Dersom
$$\lfloor l\mu \rfloor \succ_{mpo(\succsim_F)} \lfloor r\mu \rfloor$$
gjelder for alle spesialiseringer μ av alle regler $l \longrightarrow r$ i $\mathcal{R}$, så terminerer $\mathcal{R}$, gitt at $\mathcal{R}$ eller Σ er endelig eller at $\succsim_F$ er velfundert.

Figur 6.1: *Fremgangsmåten TT1-*mpo *for å vise terminering av typeordnede omskrivningssystemer.*

Bevis for korrekthet av fremgangsmåten TT1-*mpo* Kravene 2 og 3 i til TT1-*mpo* beskrevet i figur 6.1 sikrer at alle reglene i speilingen er $\succ_{mpo}$ -minskende.

Når $\mathcal{R}$ er endelig er også F endelig og $\succsim_F$ følgelig velfundert. Teorem 2.34 sier at $\succsim_{mpo}$ er velfundert når $\succsim_F$ er det. I dette tilfellet terminerer speilingen opplagt.

Anta så at Σ er uendelig og at $\succsim_F$ *ikke* er velfundert. Dessverre er R' ikke nødvendigvis endelig når $\mathcal{R}$ er det, fordi en uendelig stor Σ kan forårsake at R' i dette tilfellet inneholder uendelig mange ekstraregler. Siden $\succ_{mpo}$ er en forenklingsordning terminerer R' dersom enhver R'-omskrivningssekvens $t_1 \xrightarrow[R']{} \cdots$ er bygget opp av et endelig antall funksjonssymboler. Gitt en vilkårlig term $t_1 \in T(F)$. La så $\mathcal{F}$ være mengden av (Σ-) funksjonssymboler i t_1 og $\mathcal{R}$. Siden $\mathcal{R}$ er endelig, må også mengden $\mathcal{F}$ være endelig. La mengden $F' = \{f_w \mid f_w \in F \wedge f \in \mathcal{F}\}$. Imidlertid er ikke F' nødvendigvis endelig nå heller fordi et $\mathcal{F}$-funksjonssymbol f kan gi opphav til uendelige mange funksjonssymboler f_w siden det ikke er stilt krav om at en typesekvens w' ikke kan ha uendelige mange typesekvenser w som er *større* enn w'. Dette problemet kan løses ved å sette F'' lik alle F-symbolene i den endelige mengden av opprinnelige R'-regler i tillegg til F-symbolene i $\lfloor t_1 \rfloor$. Mengden F'' er da endelig og mengden F' kan da redefineres $\{f_{w'} \mid f_w \in F'' \wedge w' < w\}$. På denne måten er det opplagt at F' er endelig. Siden ingen ekstraregel introduserer et funksjonssymbol $f_{w'}$ der f ikke er med i $\mathcal{F}$ og $f_{w'}$ ikke er med i F', må mengden av funksjonssymboler som forekommer i en omskrivningssekvens

$$t_1 \xrightarrow[R']{} t_2 \xrightarrow[R']{} t_3 \xrightarrow[R']{} \cdots$$

(av grunntermer) være inneholdt i F'. En uendelig sekvens av termer bygget opp av denne endelige mengden funksjonssymboler ville nødvendigvis vært selvomsluttende. Men når $R' \subseteq \succ_{mpo}$ og $\succ_{mpo}$ er en forenklingsordning, er dette umulig.

I begge tilfellene terminerer R' og teorem 6.5 gir da at $\mathcal{R}$ terminerer. □

Enhver fremgangsmåte for å vise utypet terminering kan brukes av TT1-speilingen på tilsvarende måte. Man må da ha spesialtrekkene til ordningene i bakhodet. Eksempelvis må man ved bruk av $\succsim_{lpo}$ i stedet for $\succsim_{mpo}$ forlange *både* at $\succsim_F$ er velfundert *og* at alle $\approx_F$-like funksjonssymboler har samme aritet når regelmengden $\mathcal{R}$ og signaturen Σ er uendelige.

Beregnbarhet

TT1-*mpo* beskrevet over kan brukes som prosedyre for å påvise terminering av endelige systemer siden $\lfloor\ \rfloor$ er greit beregnbar, mens mengdene R' og F er endelige når $\mathcal{R}$ og Σ er endelige. Jeg har tidligere begrunnet at det er beregnbart hvorvidt et par av termer er $\succ_{mpo}$-minskende.

6.7 Begrunnelser for speilingen

Fremgangsmåten med speilinger på denne måten virker kanskje ikke så rask og det kunne være fristende å prøve å forbedre den. Jeg vil derfor begrunne min fremgangsmåte og vise det ikke er lett å finne på smarte triks som forbedrer fremgangsmåten, bortsett fra forbedringene som presenteres i de neste to kapitlene.

For å starte med funksjonssymbolsmerkingsfunksjonen $\lfloor\ \rfloor_F$, tok den utgangspunkt i argumentenes minste type. Jeg har tidligere vist at dette gir det beste resultatet.

Det er en ulempe mhp. effektivitet at vi må ha $\lfloor l\mu \rfloor \succ \lfloor r\mu \rfloor$ for alle spesialiseringer μ av variablene i l. Følgende eksempel viser nødvendigheten av dette:

Eksempel 6.11 Følgende system terminerer ikke:

Typer: s, s_1, s_2, s_{12}
$s_1, s_2 \leq s$
$s_{12} \leq s_1$
$s_{12} \leq s_2$
Funksjoner: $f : s_1 \rightarrow s$
$f : s_2 \rightarrow s$
$f : s_{12} \rightarrow s$
$g : s_1 \rightarrow s_2$
$g : s_{12} \rightarrow s_{12}$
$a : s_{12}$
Regel: $(\forall x : s_1)\ \ f(x) \longrightarrow f(g(x))$

Merk at symbolet $f_{s_{12}}$ *må* være med i funksjonsymbolmengden F fordi $\lfloor\ \rfloor_F$ ellers ikke ville vært veldefinert. Anta nå at vi *ikke* sjekker for alle mulige spesialiseringer. Da ville vi slått fast terminering av regel $f_{s_1}(x) \longrightarrow f_{s_2}(g_{s_1}(x))$, som i det forrige eksemplet. Det typeordnede systemet terminerer imidlertid ikke, siden det kan ha en uendelig avledning $f(a) \xrightarrow[\mathcal{R}]{} f(g(a)) \xrightarrow[\mathcal{R}]{} f(g(g(a))) \xrightarrow[\mathcal{R}]{} \cdots$. Dersom vi her hadde sjekket for alle spesialiseringen hadde vi for $\mu = \{x \mapsto x : s_{12}\}$ fått $\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$ som gir at vi måtte vise $f_{s_{12}}(x) \succ f_{s_{12}}(g_{s_{12}}(x))$, som ikke gjelder for noen fornuftig termineringsordning $\succ$.

At man må vise $\lfloor l\mu \rfloor \succ_{mpo} \lfloor r\mu \rfloor$ for alle spesialiseringer gjør testen på terminering litt langsom. Likevel er ikke denne kompleksiteten verre enn man kunne forvente, da man også må se på alle spesialiseringer for å avgjøre egenskaper som konfluens og typeminskning. Denne kompleksiteten synes derfor innebygget i problemet. En annen grunn til at dette ikke er så dramatisk er at det å avgjøre hvorvidt en regel er $\succ_{mpo}$ -minskende er såpass tidkrevende i seg selv at utvidelsen i de fleste tilfellene ikke gjør termineringstesten signifikant langsommere.

Man kan heller ikke bare sjekke en variabels "laveste" spesialisering(er), som jeg her illustrerer.

Eksempel 6.12 Hvis man i det ikke-terminerende systemet

Typer: s, s'
$s' \leq s$
Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s'$
$g : s \rightarrow s$
$g : s' \rightarrow s'$
$a : s$
Regler: $g(a) \longrightarrow f(a)$
$(\forall x : s)\ \ f(x) \longrightarrow g(x)$

bare ser på den laveste spesialiseringen av x, dvs. at x får typen s', så får vi at vi må vise $g_s(a) \succ_{mpo} f_s(a)$ og $f_{s'}(x) \succ_{mpo} g_{s'}(x)$ og det klarer vi å gjøre ved presedensen $g_s \succ_F f_s \succ_F f_{s'} \succ_F g_{s'}$ selv om systemet ikke terminerer siden det fins en uendelig sekvens $g(a) \xrightarrow[\mathcal{R}]{} f(a) \xrightarrow[\mathcal{R}]{} g(a) \xrightarrow[\mathcal{R}]{} f(a) \xrightarrow[\mathcal{R}]{} \cdots$.

Her så vi at det var konstant av en "høyere" type ($s \geq s'$) som gjorde at vi ikke bare kunne sjekke laveste spesialiseringer, så da kunne det kanskje være en idé å "senke" typen på konstanter. Dette ville imidlertid føre til mer jobb i noen tilfeller, da en type s kan ha flere minste typer $s_1, s_2, \ldots$ som alle er mindre enn s og hvor disse ikke har noen type under seg. Så $f(a : s)$ ville da blitt til $f(a : s_1), f(a : s_2)$ osv. Denne løsningen ville også nærme seg et utypet system, slik at speilingen ville bli mindre nøyaktig i termineringshenseende, noe følgende eksempel viser.

Eksempel 6.13 Det typeordnede systemet

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \longrightarrow s$
	$f : s' \longrightarrow s'$
	$g : s \longrightarrow s$
	$g : s' \longrightarrow s'$
	$a : s$
Regler:	$g(a) \longrightarrow f(a)$
	$(\forall x : s')\ \ f(x) \longrightarrow g(x)$

terminerer, noe som også kan vises ved TT1, men dersom vi setter typen til a til s' istedet kan algoritmen ikke påvise terminering da speilingen R' ville fått en uendelig sekvens $f_{s'}(a) \xrightarrow[R']{} g_{s'}(a) \xrightarrow[R']{} f_{s'}(a) \xrightarrow[R']{} \cdots$

Vi må altså innse at speilingen R' må inneholde en regel $\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$ for enhver regelspesialisering $l\mu \longrightarrow r\mu$ i $\mathcal{R}$. Spørsmålet er om det virkelig er nødvendig å *vise* $\lfloor l\mu \rfloor \succ_{mpo} \lfloor r\mu \rfloor$ for alle disse. Man kunne tenke at $\lfloor l\mu \rfloor \succ_{mpo} \lfloor r\mu \rfloor$ fulgte fra $\lfloor l \rfloor \succ_{mpo} \lfloor r \rfloor$ med passende tilleggsbetingelser til presedensen $\succsim_F$. Det viser seg at det ikke forholder seg på denne måten. Den ene grunnen er at slike tilleggskrav ville gjøre ordningene meget svake. Klarer vi å vise $\lfloor l \rfloor = f_w(t_1, \ldots, t_m) \succ_{mpo} g_{w'}(t'_1, \ldots, t'_n) = \lfloor r \rfloor$ ved hjelp av presedensen $f_w \succ_F g_{w'}$, må vi forlange $f_{w_1} \succ_F g_{w'_1}$ for alle par $f_{w_1}, g_{w'_1}$ i F der $w_1 \leq w$ og $w'_1 \leq w'$. Dette ville svekke ordningen betraktelig. Men ikke engang slike krav hjelper, fordi selv om vi viser

$$\lfloor l \rfloor = f_w(t) \succ_{mpo} f_w(t') = \lfloor r \rfloor$$

ved tilfelle *mpo-3*, kan det finnes en spesialisering μ der $\lfloor l\mu \rfloor = f_{w'}(t^*)$ og $\lfloor r\mu \rfloor = f_{w''}(t'^*)$. Vi har i dette tilfellet ingen antagelser om forholdet mellom $f_{w'}$ og $f_{w''}$ slik at

$$\lfloor l\mu \rfloor = f_{w'}(t^*) \succ_{mpo} f_{w''}(t'^*) = \lfloor r\mu \rfloor$$

ikke nødvendigvis gjelder. Man må derfor (hvert fall i ordningen $\succsim_{mpo}$) eksplisitt *vise* $\lfloor l\mu \rfloor \succ_{mpo} \lfloor r\mu \rfloor$ for alle regelspesialiseringer.

Det hadde også vært ønskelig å tjene inn noe på selve ordningen, ikke bare betrakte den som en ren utypet ordning over speilinger, men både spise opp kaken og likevel beholde den. Jeg tenkte på å kunne utnytte forskjellen i ariteter og samtidig beholde likhet over funksjonssymboler. Kort sagt, bruke $f_{s_1} \succ_F f_{s_2}$ i tilfelle to av *mpo*-definisjonen, samtidig som man i andre tilfeller kunne si at alle f'ene er like når dét passer seg. Ikke uventet viser følgende eksempel at det ikke lar seg gjøre.

Eksempel 6.14 Det er lett å se at systemet

Typer:	s, s_1, s_2
	$s_1, s_2 \leq s$
Funksjoner:	$f : s_1 \rightarrow s$
	$f : s_2 \rightarrow s$
	$a : s_1$
	$b : s_2$
Regler:	$f(a) \longrightarrow f(b)$
	$f(b) \longrightarrow f(a)$

ikke terminerer. Speilingen ville gi $f_{s_1}(a) \longrightarrow f_{s_2}(b)$ og $f_{s_2}(b) \longrightarrow f_{s_1}(a)$. Dersom man på én eller annen måte kunne ha både $f_{s_1} \succ_F f_{s_2}$ og $f_{s_1} \approx_F f_{s_2}$, ville man kunne vise $\succ_{mpo}$ -minskning av begge reglene, uten at regelsettet terminerer. Man kan altså ikke både "splitte" et funksjonssymbol og likevel ikke splitte det.

Det synes altså vanskelig å gjøre de helt store forbedringene i TT1-fremgangsmåten. De forbedringene jeg ser for meg, presenteres i de to neste kapitlene, men grunnidéen vil forbli den samme.

6.8 En svakhet med TT1

Jeg vil senere komme tilbake til en helhetsvurdering av hele TT-metoden.

En alvorlig svakhet ved TT1-*mpo* er at den faktisk av og til kan være dårligere enn TT0-*mpo*, dvs. vanlig $\succsim_{mpo}$ der vi ikke filosoferer over typene. Følgende eksempel viser dette:

Eksempel 6.15 La det terminerende omskrivningssystemet $\mathcal{R}$ være

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s$
	$a : s'$
	$b : s$
Regler:	$f(a) \longrightarrow f(b)$
	$(\forall x : s')\ \ f(x) \longrightarrow a$

Bruker man utypet $\succsim_{mpo}$ -analyse, kan terminering vises enkelt med presedensen $f \succ_F a \succ_F b$. TT1-speiles $\mathcal{R}$, får man regelsettet $R' = \{f_{s'}(a) \longrightarrow f_s(b),\ f_{s'}(x) \longrightarrow a\}$. Fordi $s' < s$ så forlanger TT1-*mpo* $f_s \succ_F f_{s'}$, og for å vise terminering av den første regelen må $a \succ_F f_s$ og $a \succ_F b$. For å vise terminering av den andre regelen må vi ha $f_{s'} \succ_F a$, noe som er umulig da $f_s \succ_F f_{s'}$ og $a \succ_F f_s$ gir oss $a \succ_F f_{s'}$.

Det var kanskje ikke så lurt å se på hvert "funksjonssymbol med aritet" som et distinkt symbol likevel? Dette skal jeg rette på i utvidelsen TT2, der vi kan gruppere sammen f_s'er som stammer fra samme Σ-symbol f.

Kapittel 7

TT2-metoden – en forbedring av TT1-metoden

Speilingen i fremgangsmåten TT1 inneholdt en ekstraregel $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ for alle par av funksjonssymboler $f_w, f_{w'} \in F$ der $w' < w$. Presedensen $\succsim_F$ måtte oppfylle $f_w \succ_F f_{w'}$ for at denne ekstraregelen skulle være $\succ_{mpo}$ -minskende. Eksempel 6.15 viste at dette var en kraftig restriksjon, da selv en utypet analyse kunne gi bedre resultater enn TT1-*mpo*.

Dette kapitlet presenterer fremgangsmåten TT2 som er en enkel men kraftig forbedring av TT1. I TT2 kan man la funksjonssymbolene f_w og $f_{w'}$ være "ekvivalente" i ordningene selv om $w' < w$.

Dette kan sees på som at man "grupperer sammen" symboler som skal ha lik "vekt" innen ordningene. Samme effekt kan fås ved å fjerne deklarasjoner fra signaturen Σ og la den utypede funksjonssymboplmengden inneholde ett funksjonssymbol for hver nye signaturen Σ. En slik variant er mer komplisert siden man hele tiden må passe på at Σ er regulær.

Jeg vil bruke ordningen *mpo* for å illustrere fremgangsmåten TT2, som er en ren utvidelse av TT1 og bygger på samme idéer og begreper som denne. Derfor er funksjonssymbolmengden F den samme mengden, og $\lfloor\ \rfloor$ og $\lfloor\ \rfloor_F$ de samme funksjonene som ble definert i forrige kapittel.

7.1 TT2-*mpo*

Eneste forskjellen mellom TT1-*mpo* og TT2-*mpo* er at TT1-*mpo*-kravet

$$\forall f_w, f_{w'} \in F \mid w' < w \Longrightarrow f_w \succ_F f_{w'}$$

erstattes med det mindre strenge kravet

$$\forall f_w, f_{w'} \in F \mid w' < w \Longrightarrow f_w \succsim_F f_{w'} \tag{7.1}$$

Dette betyr at vi kan ha $f_w \approx_F f_{w'}$ selv om $w' < w$, men betyr også at ekstraregelen $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ ikke lenger er $\succ_{mpo}$ -minskende. Problemet løses ved å

definere en annen speiling, der alle $f_w, f_{w'}$ som er $\approx_F$-like representeres av samme funksjonssymbol. Den nye speilingen trenger man ikke tenke på ved bruk av TT2-*mpo*, slik at fremgangsmåten TT2-*mpo* for å påvise terminering av typeordnede systemer blir som beskrevet i figur 7.1, der Σ som vanlig er en preregulær signatur og $\mathcal{R}$ en typeminskende regelmengde.

1. Definér den utypede funksjonssymbolmengden F og funksjonssymbolsmerkingsfunksjonen $\lfloor\ \rfloor_F$ på én av måtene beskrevet i avsnitt 6.3. Jeg ville anbefale å definere F ved
$$F = \{f_{w'} \mid \exists s | f \in \Sigma_{w \to s} \wedge w' \leq w\}$$
2. La $\succsim_F$ være en kvasiordning over F som tilfredsstiller
$$f_w, f_{w'} \in F \wedge w' < w \Longrightarrow f_w \succsim_F f_{w'}$$
3. Dersom
$$\lfloor l\mu \rfloor \succ_{mpo(\succsim_F)} \lfloor r\mu \rfloor$$
gjelder for alle spesialiseringer μ av alle regler $l \longrightarrow r$ i $\mathcal{R}$, så terminerer $\mathcal{R}$, gitt at $\mathcal{R}$ eller Σ er endelig eller at $\succsim_F$ er velfundert.

Figur 7.1: *Fremgangsmåten TT2-mpo for å vise terminering av et typeordnet omskrivningssystem* $\langle \Sigma, \mathcal{R} \rangle$.

Bevis for korrekthet av fremgangsmåten TT2-*mpo*: Beviset for at $\mathcal{R}$ terminerer når alle krav i TT2-*mpo*-metoden er oppfylt kommmer som eget avsnitt, avsnitt 7.2. □

For å lette forståelsen og for å kunne bruke TT2 også for ordninger som bygger på en ikke-refleksiv presedens $\succ$ over funksjonssymboler, følger følgende to påstander fra krav 7.1:
$$\forall f_s, f_{s'} \in F \mid s' < s \Longrightarrow f_s \succ_F f_{s'} \vee f_s \approx_F f_{s'} \tag{7.2}$$
og
$$\forall f_s, f_{s'}, f_{s''} \in F \mid s'' < s' < s \wedge f_{s''} \approx_F f_s \Longrightarrow f_{s''} \approx_F f_{s'} \approx_F f_s \tag{7.3}$$
Påstand 7.2 er opplagt ut i fra definisjon av kvasiordninger og krav 7.1.

Bevis for 7.3: Anta $f_{s''} \approx_F f_s$. Krav 7.1 gir $f_s \succsim_F f_{s'}$ og $f_{s'} \succsim_F f_{s''}$. Definisjonen av kvasiordninger sier at $f_{s''} \approx_F f_s$ betyr $f_{s''} \succsim_F f_s$ og $f_s \succsim_F f_{s''}$. Transitivitet av $\succsim_F$, samt $f_{s'} \succsim_F f_{s''}$ og $f_{s''} \succsim_F f_s$ gir $f_{s'} \succsim_F f_s$. Fra før av hadde vi $f_s \succsim_F f_{s'}$, slik at disse to siste gir oss $f_s \approx_F f_{s'}$. Jeg har tidligere vist at $\approx_F$ er en ekvivalensrelasjon, slik at $f_{s''} \approx_F f_s$ og $f_s \approx_F f_{s'}$ gir oss ønskede $f_{s''} \approx_F f_s \approx_F f_{s'}$. □

TT2-*mpo* er en ekte utvidelse av TT1-*mpo* (fordi TT2-*mpo* er lik TT1-*mpo* hvis $f_w \succ_F f_{w'}$ når $w' < w$) og TT0-*mpo* (siden TT2-*mpo* med presedensen $f_w \approx_F f_{w'}$ for alle $f_w, f_{w'} \in F$ er

lik TT0-*mpo*), mens TT1 som vi har sett ikke var en ekte utvidelse av TT0. Med TT2-*mpo* kan man vise terminering av systemet i eksempel 6.15 med presedensen $f_s \approx_F f_{s'} \succ_F a \succ_F b$. Følgende eksempel viser at TT2-*mpo* kan vise terminering av et system som hverken TT0-*mpo* (en utypet *mpo*-analyse) eller TT1-*mpo* kan vise terminerer:

Eksempel 7.1 Omskrivningssystemet

Typer: s, s_1, s_2, s_{11}
$s_1, s_2 \leq s$
$s_{11} \leq s_1$

Funksjoner: $f : s \rightarrow s$
$f : s_1 \rightarrow s$
$f : s_2 \rightarrow s$
$f : s_{11} \rightarrow s$
$g : s_1 \rightarrow s_2$
$a : s_1$
$b : s_{11}$

Regler: $(\forall x : s_1)\ f(x) \longrightarrow f(g(x))$
$f(b) \longrightarrow f(a)$
$(\forall y : s_{11})\ f(y) \rightarrow b$

terminerer, noe som på grunn av den første regelen ikke kan vises ved en utypet (TT0-) analyse.

Følgende må vises for speilingen R':

$$\begin{array}{ll} f_{s_1}(x) \succ_{mpo} f_{s_2}(g_{s_1}(x)) & \\ f_{s_{11}}(x) \succ_{mpo} f_{s_2}(g_{s_1}(x)) & \text{(fra spesialiseringen } \mu = \{x : s_1 \mapsto x : s_{11}\}) \\ f_{s_{11}}(b) \succ_{mpo} f_{s_1}(a) & \\ f_{s_{11}}(y) \succ_{mpo} b & \end{array}$$

TT1-*mpo* forlanger her $f_{s_1} \succ_F f_{s_{11}}$. For å vise $f_{s_{11}}(y) \succ_{mpo} b$ må vi ha $f_{s_{11}} \succsim_F b$, tilsammen gir dette $f_{s_1} \succ_F f_{s_{11}} \succsim_F b$, dermed kan ikke $f_{s_{11}}(b) \succ_{mpo} f_{s_1}(a)$ vises, slik at TT1-*mpo* ikke kan vise terminering av eksemplet.

Bruker vi imidlertid TT2-*mpo* gir for eksempel presedensen

$$f_s \succ_F f_{s_1} \approx_F f_{s_{11}} \succ_F f_{s_2} \approx_F g_{s_1} \approx_F b \succ_F a$$

at de 4 reglene i speilingen gitt over er $\succ_{mpo}$-minskende samtidig som krav 7.1 er tilfredsstilt. Vi har dermed vist terminering av det typeordnede systemet.

7.2 Korrekthetsbevis for TT2-*mpo*-metoden

Jeg skal i dette avsnittet bevise at TT2-*mpo*-metoden er korrekt, dvs. at $\mathcal{R}$ virkelig terminerer dersom alle betingelsene til TT2-*mpo*-metoden i fig. 7.1 er oppfylt. For å få til dette vil jeg vise at det fins en speiling $\langle F, \lfloor\ \rfloor^*, R'' \rangle$ av et typeordnet system $\langle \Sigma, \mathcal{R} \rangle$ som terminerer når TT2-*mpo*-betingelsene holder. Forskjellen fra TT1-speilingen er at $\approx_F$-like funksjonssymboler blir slått sammen i den nye speilingen R'', og at alle ekstraregler på formen $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ fjernes dersom $f_w \approx_F f_{w'}$. Beviset vil innføre en ny termspeilingsfunksjon $\lfloor\ \rfloor^*$ som bare brukes her. *Bruken* av TT-metodene vil som nevnt hele tiden basere seg på termspeilingsfunksjonen $\lfloor\ \rfloor$ beskrevet i forrige kapittel.

7.2.1 Speilingen

I dette avsnittet skal jeg definere en speiling $\langle F, \lfloor\ \rfloor^*, R''\rangle$ og vise at den virkelig er en speiling av et typeordnet system $\langle \Sigma, \mathcal{R}\rangle$.

Funksjonssymbolmengden F

Den utypede funksjonssymbolmengden F vil være den samme som beskrevet i avsnitt 6.3 for TT1-metoden.

Termspeilingsfunksjonen $\lfloor\ \rfloor^*$

Jeg har tidligere vist at $\approx_F$ indusert av en kvasiordning $\succsim_F$ er en ekvivalensrelasjon og dermed danner ekvivalensklasser over F. Hovedidéen bak TT2-speilingen er at $\approx_F$-like $f_w, f_{w'}, f_{w''}, \ldots$ skal representeres i speilingen av ett (felles) funksjonssymbol, dvs. av ett element i $\approx_F$-ekvivalensklassen $[f_w]_{\approx_F}$. Dersom $[f_w]_{\approx_F}$ er en ekvivalensklasse, så kan den imidlertid inneholde både f'er og andre funksjonssymboler fra Σ, f. eks. g. De utypede symbolene f_w og $g_{w'}$ kan ha forskjellig aritet. Noen reduksjonsordninger, som bl. a. *lpo*, er generelt ikke definert for utypede systemer der et funksjonssymbol kan ha variabel aritet, så vi må være litt forsiktige med å slå sammen funksjonssymboler med forskjellig aritet til ett symbol. For å være på den sikre siden, og for å ha et mer generelt bevis, definerer jeg en *kanonisk representant* utifra funksjonssymboler i Σ. Det betyr for at for enhver $f \in \Sigma$ velger jeg en f-kanonisk representant, kalt f-crep$([f_w]_{\approx_F})$ i $[f_w]_{\approx_F}$.

For enhver ekvivalensklasse og hvert funksjonssymbol i Σ (som er representert i ekvivalensklassen) finnes altså en f-crep$([f_w]_{\approx_F}) \in F$. Denne f-kanoniske representanten for $[f_w]_{\approx_F}$ velger jeg å kalle ${f_w}^*$. Dersom $[f_w]_{\approx_F}$ bare inneholder ett element som stammer fra en f i Σ, skriver jeg av og til bare f_w for den éntydige ${f_w}^*$[1]. For å sammenfatte disse nye begrepene er altså ${f_w}^* = f$-crep$([f_w]_{\approx_F})$ et funksjonssymbol $f_{w'}$ i F der $f_w \approx_F f_{w'}$ og der f_w og $f_{w'}$ har lik aritet.

Eksempel (forts. av eks. 7.1) Ekvivalensklassen $[f_{s_2}]_{\approx_F}$ indusert av presedensen $\succsim_F$ definert tidligere i eksempel 7.1 består av elementene f_{s_2}, g_{s_1} og b. Her vil da f-crep$([f_{s_2}]_{\approx_F}) = f_{s_2}$, g-crep$([f_{s_2}]_{\approx_F}) = g_{s_1}$ og b-crep$([f_{s_2}]_{\approx_F})$ vil være lik b. Tilsvarende kunne vi ha f-crep$([f_{s_{11}}]_{\approx_F}) = {f_{s_{11}}}^* = f_{s_1}$.

La nå den nye termspeilingsfunksjonen $\lfloor\ \rfloor^* : \mathcal{T}(\Sigma, X) \rightarrow T(F, V)$ være definert ved at $\lfloor t \rfloor^*$ er lik $\lfloor t \rfloor$ med den forskjellen at ethvert symbol f_w i t blir erstattet med symbolet ${f_w}^*$. Mer formelt er en funksjonssymbolsmerkingsfunksjon $\lfloor\ \rfloor_F^* : \Sigma \times S^* \rightarrow F$ definert ved

$$\lfloor f, w \rfloor_F^* = f\text{-crep}([\lfloor f, w \rfloor_F]_{\approx_F})$$

der $\lfloor\ \rfloor_F$ er som definert i forrige kapittel. Funksjonen $\lfloor\ \rfloor^*$ er på vanlig måte utvidelsen av $\lfloor\ \rfloor_F^*$ til en termspeilingsfunksjon.

[1] Er man meget pirkete kan f ta ulikt antall argumenter dersom $f : s \rightarrow s$ og $f : s, s \rightarrow s$ er to deklarasjoner i Σ. Hvis nå $f_s \approx_F f_{s,s}$ vil ${f_s}^*$ ta enten ett eller to argumenter. Dette kan bøtes på ved å definere f_s^* som et F-symbol som tar ett argument og $f_{s,s}^*$ som et annet F-symbol som tar to argumenter. Dette vil ikke forårsake problemer da F-sprang fra f_s til $f_{s',s'}$ eller omvendt aldri kan forekomme.

Eksempel (forts. av eks. 7.1) Den eneste forskjellen mellom $\lfloor t \rfloor$ og $\lfloor t \rfloor^*$ ville være at enhver forekomst av $f_{s_{11}}$ i $\lfloor t \rfloor$ ville bli erstattet med f_{s_1} i $\lfloor t \rfloor^*$. Eksempelvis er $\lfloor f(b) \rfloor = f_{s_{11}}(b)$ mens $\lfloor f(b) \rfloor^* = f_{s_1}(b)$.

Regelmengden R''

Den speilende regelmengden R'' består som før av to mengder regler. De "opprinnelige" reglene blir som for TT1, dvs. for alle regler $l \longrightarrow r$ i $\mathcal{R}$ og for alle spesialiseringer μ for variablene i l er $\lfloor l\mu \rfloor^* \longrightarrow \lfloor r\mu \rfloor^*$ med i R''.

Ekstrareglene i R' beholdes også, unntatt de som i det tilfellet ville gitt en regel som ikke gjør noe som helst. Formalisert vil det si at mengden av ekstraregler i R'' er gitt ved

$$R'' = \{ {f_w}^*(x_1, \ldots, x_n) \longrightarrow {f_{w'}}^*(x_1, \ldots, x_n) \mid f_w, f_{w'} \in F \wedge s' < s \wedge {f_w}^* \neq {f_{w'}}^* \}$$

R'' er altså mengden av de opprinnelige reglene og ekstrareglene.

Eksempel (forts. av eks. 7.1) Hvis vi definerer $f\text{-crep}([f_{s_1}]_{\approx_F}) = f_{s_1} = {f_{s_1}}^* = {f_{s_{11}}}^*$ ville den speilende regelmengden R'' til systemet i eksempel 7.1 blitt som følger:

$$\begin{array}{ll}
f_{s_1}(x) \longrightarrow f_{s_2}(g_{s_1}(x)) & \\
f_{s_1}(x) \longrightarrow f_{s_2}(g_{s_1}(x)) & \text{(fordi } {f_{s_{11}}}^* = f_{s_1}) \\
f_{s_1}(b) \longrightarrow f_{s_1}(a) & \text{(også fordi } {f_{s_{11}}}^* = f_{s_1}) \\
f_{s_1}(y) \longrightarrow b & \text{(tilsvarende her)}
\end{array}$$

Disse er de jeg kaller de opprinnelige reglene i R''. Ekstrareglene er da

$$\begin{array}{l}
f_s(x) \longrightarrow f_{s_1}(x) \\
f_s(x) \longrightarrow f_{s_2}(x)
\end{array}$$

Her kan man legge merke til at ekstraregelen $f_{s_1}(x) \longrightarrow f_{s_{11}}(x)$ som vi ville hatt i en TT1-speiling forsvinner. Med presedensen $\succsim_F$ gitt tidligere kan man med hjelp av *mpo* eller *lpo* vise at regelmengden R'' terminerer, noe jeg snart vil vise også impliserer terminering av $\mathcal{R}$.

Dersom t er en utypet term, vil jeg skrive t^* for termen som fremkommer når alle funksjonssymboler f_w i t erstattes med tilhørende f-kanoniske representanten ${f_w}^*$.

Det må nå vises at $\langle F, \lfloor\ \rfloor^*, R'' \rangle$ virkelig er en speiling av et typeordnet omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$.

Teorem 7.1 $\forall t, t' \in \mathcal{T}(\Sigma, X) \mid t \xrightarrow[\mathcal{R}]{} t' \Longrightarrow \lfloor t \rfloor^* \xrightarrow[R'']{+} \lfloor t' \rfloor^*$

Bevis: Beviset tar utgangspunkt i speilingen $\langle F, \lfloor\ \rfloor, R' \rangle$ vi brukte i TT1-beviset. Husk at $\lfloor t \rfloor^*$ bare er $\lfloor t \rfloor$ med alle funksjonssymboler f_w byttet ut med ${f_w}^*$.

Vi hadde i beviset for teorem 6.15 en omskrivningssekvens

$$\lfloor t \rfloor \xrightarrow[R']{opp} t_1 \xrightarrow[R']{eks} t_2 \cdots \xrightarrow[R']{eks} t_n = \lfloor t' \rfloor$$

der $n \geq 1$, og der symbolet $\xrightarrow[R']{opp}$ står for bruk av én av de "opprinnelige" reglene i R' og $\xrightarrow[R']{eks}$ står for bruk av en ekstraregel. Siden vi i R'' ikke har fjernet noen av de opprinnelige reglene i R' og fordi enhver f_w i både $\lfloor t \rfloor$ og R'_{opp} erstattes med en $f_w{}^*$ i $\lfloor t \rfloor^*$ og R''_{opp} gjelder

$$\lfloor t \rfloor^* \xrightarrow[R'']{opp} t_1{}^*$$

Det er opplagt at dersom $t_i \xrightarrow[R']{eks} t_{i+1}$ så er enten $t_i^* \xrightarrow[R'']{eks} t_{i+1}{}^*$ eller $t_i^* = t_{i+1}{}^*$ for alle termer $t_i, t_{i+1} \in T(F,V)$. Dette kommer av vi bare har fjernet ekstraregler som ikke forandrer på noe som helst i R''. Dermed kan ovenstående sekvens bli til

$$\lfloor t \rfloor^* \xrightarrow[R'']{opp} t_1{}^* \xrightarrow[R'']{eks\ *} t_2{}^* \cdots \xrightarrow[R'']{eks\ *} t_n{}^* = \lfloor t' \rfloor^*$$

Siden vi har $\lfloor t \rfloor^* \xrightarrow[R'']{opp} t_1{}^*$ inneholder ovenstående sekvens minst ett omskrivningssteg og kan derfor skrives på den ønskede formen

$$\lfloor t \rfloor^* \xrightarrow[R'']{+} \lfloor t' \rfloor^*$$

□

7.2.2 Terminering av $\mathcal{R}$

Siden $\langle F, \lfloor\ \rfloor^*, R'' \rangle$ er en speiling av $\langle \Sigma, \mathcal{R} \rangle$ er det tilstrekkelig å vise at R'' terminerer når TT2-*mpo*-betingelsene i fig. 7.1 er oppfylt for å fastslå terminering $\mathcal{R}$.

Følgende lemma, som såvidt er nevnt (uten bevis) i kapittel 2, vil vise seg være nyttig:

Lemma 7.2 $\forall t \in T(F,V) \mid t \approx_{mpo} t^*$

Bevis: Beviset er ved induksjon på oppbygning av termen t.

- t **variabel:** Dersom $t = x$ er også $t^* = x$ og $x \succsim_{mpo} x$ følger av regel *mpo-3* siden $x \approx_F x$ og $\emptyset \succsim_{mpo}^{multi} \emptyset$ ved regel *multi-1*.
- t **konstant eller sammensatt term:** Anta at $t = f(t_1, \ldots, t_n)$ der $n \geq 0$. Da er $t^* = f^*(t_1{}^*, \ldots, t_n{}^*)$ og induksjonshypotesen gir oss $t_i \approx_{mpo} t_i^*$ for alle i. Dersom $n > 0$ gir *multi-2* brukt n ganger at $\{t_1, \ldots, t_n\} \succsim_{mpo}^{multi} \{t_1{}^*, \ldots, t_n{}^*\}$ og n nye tilfeller av *multi-2* gir $\{t_1{}^*, \ldots, t_n{}^*\} \succsim_{mpo}^{multi} \{t_1, \ldots, t_n\}$. Hvis $n = 0$ er $\emptyset \succsim_{mpo}^{multi} \emptyset$. Siden $f \approx_F f^*$ gir regel *mpo-3* både $f(x_1, \ldots, x_n) \succsim_{mpo} f^*(t_1{}^*, \ldots, t_n{}^*)$ og $f^*(t_1{}^*, \ldots, t_n{}^*) \succsim_{mpo} f(t_1, \ldots, t_n)$ som gir ønskede $t \approx_{mpo} t^*$.

□

Det er da trivielt å vise

Lemma 7.3 *For alle termer t, t' i $T(F,V)$ gjelder*

$$t \succ_{mpo} t' \Longrightarrow t^* \succ_{mpo} t'^*$$

hvor t^* er termen t der alle funksjonssymboler f_w er erstattet med $f_w{}^*$.

Bevis: Følger trivielt av lemma 7.2 og egenskaper ved kvasiordninger siden $t^* \approx_{mpo} t \succ_{mpo} t' \approx_{mpo} t'^*$. □

Jeg kan nå vise korrekthet av TT2-*mpo*-metoden.

Teorem 7.4 (Korrekthet av TT2-*mpo*) *Gitt en preregulær typeordnet signatur Σ der ingen type har et uendelig antall typer under seg. Anta at $\mathcal{R}$ er et typeminskende omskrivningssystem i Σ og at F og $\lfloor\ \rfloor$ er definert som i kapittel 6. La så $\succsim_F$ være en kvasiordning over F slik at $w' < w \wedge f_w, f_{w'} \in F \Longrightarrow f_w \succsim_F f_{w'}$ og anta at $\lfloor l\mu \rfloor \succ_{mpo(\succsim_F)} \lfloor r\mu \rfloor$ gjelder for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$.*

Da terminerer $\mathcal{R}$ dersom enten $\mathcal{R}$ eller Σ er endelig eller $\succsim_F$ er velfundert.

Bevis: Dersom R'' beskrevet tidligere i dette kapittel terminerer, gir teorem 6.5 at $\mathcal{R}$ terminerer.

R'' består av en mengde "opprinnelige" regler $\{\lfloor l\mu \rfloor^* \longrightarrow \lfloor r\mu \rfloor^* \mid l\mu \longrightarrow r\mu$ regelspesialisering i $\mathcal{R}\}$ og en mengde "ekstraregler" $\{f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n) \mid w' < w \wedge f_w, f_{w'} \in F \wedge f_w{}^* \neq f_{w'}{}^*\}$. Kravet $\lfloor l\mu \rfloor \succ_{mpo(\succsim_F)} \lfloor r\mu \rfloor$ og lemma 7.3 gir $\lfloor l\mu \rfloor^* \succ_{mpo(\succsim_F)} \lfloor r\mu \rfloor^*$ for alle opprinnelige regler i R'' og kravene $f_w \succsim_F f_{w'}$ og $f_w{}^* \neq f_{w'}{}^*$ gir $f_w \succ_F f_{w'}$ og følgelig $f_w(x_1, \ldots, x_n) \succ_{mpo(\succsim_F)} f_{w'}(x_1, \ldots, x_n)$ for alle ekstrareglene i R''. Alle reglene i R'' er dermed inneholdt i $\succ_{mpo(\succsim_F)}$. Hvis Σ er endelig, er også F endelig og siden $\succ_{mpo(\succsim_F)}$ er en forenklingsordning så terminerer R''. Når $\succsim_F$ er velfundert er også $\succ_{mpo(\succsim_F)}$ velfundert og da terminerer også R''. I det siste tilfellet, nemlig at $\mathcal{R}$ er endelig, er enhver sekvens $t_1 \xrightarrow[R'']{} t_2 \xrightarrow[R'']{} \cdots$ av grunntermer bygget opp av en endelig mengde funksjonssymboler[2], slik at en uendelig sekvens av disse termene ville vært selvomsluttende. Denne selvomsluttende sekvensen kan ikke finnes da $\xrightarrow[R'']{} \subseteq \succ_{mpo(\succsim_F)}$.

I alle tre tilfellene terminerer R'' og følgelig også $\mathcal{R}$. □

7.3 Generalisering av TT2

Metoden TT2-*mpo* kan generaliseres til å kunne brukes på alle utypede termineringsordninger $\succ$, ved at man gir symbolene f_w og $f_{w'}$ lik "vekt/verdi" innenfor ordningen. Man må da huske på at alle F-funksjonssymbolene $f_{w''}$ der $w' < w'' < w$ også blir tilordnet denne samme "vekten". Når $w' < w$ og symbolene f_w og $f_{w'}$ *ikke* har samme vekt, må man også sikre at ekstraregelen $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ er $\succ$-minskende.

Det kan forøvrig bemerkes at forandringen fra TT1 til TT2 tok utgangspunkt i ulemper ved stiordningene. Speilingen $\langle F, \lfloor\ \rfloor^*, R'' \rangle$ er faktisk svakere enn TT1-speilingen $\langle F, \lfloor\ \rfloor, R' \rangle$ i termineringshenseende siden R' alltid terminerer når R'' gjør det. For andre klasser av ordninger kan det derfor være aktuelt med andre forbedringer av TT1-speilingen, men TT2-metoden er alltid minst like sterk som TT0- og TT1-metodene for alle ordninger.

[2] noe som kan vises på samme måte som for TT1

Kapittel 8

TT3-metoden – en utvidelse av TT2-metoden

TT2-*mpo* var som vi har sett en klar forbedring av TT1-*mpo*, men fremdeles hadde den kravet om at dersom $w' < w$, så måtte presedensen $\succsim_F$ oppfylle $f_w \succsim_F f_{w'}$. Av og til kan det imidlertid være nødvendig å bruke en presedens der $f_{w'} \succ_F f_w$ for å kunne påvise terminering, slik at krav 7.1 må brytes. Dette fører dessverre til mange ekstra vanskeligheter, som jeg vil beskrive i dette kapittel.

TT3-*mpo* kommer også ha den svakheten at vi bare kan tillate $f_{w'} \succ_F f_w$ (antatt $w' < w$) dersom f er *unær*, dvs. bare tar ett argument. Denne svakheten gjelder alle stiordningene, men jeg vil vise lenger ut i oppgaven at det fins termineringsordninger som ikke har denne begrensningen.

8.1 Motivasjon og vanskeligheter

La se på følgende eksempel én gang til:

Eksempel 8.1 Omskrivningssystemet

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s$
	$g : s' \rightarrow s$
Regel:	$(\forall x : s')\ \ f(x) \longrightarrow f(g(x))$

terminerer, da $g(x)$ har typen s, slik at denne *ikke* kan substitueres for $x : s'$. Man kan imidlertid ikke vise terminering av regelen med noen av metodene vi har sett på hittil, hverken med en utypet eller typet analyse. Regelen speiles i $f_{s'}(x) \longrightarrow f_s(g_{s'}(x))$. Det kunne vært mulig å vise $f_{s'}(x) \succ_{mpo} f_s(g_{s'}(x))$ med f. eks. presedensen $f_{s'} \succ_F f_s \succ_F g_{s'}$. Denne presedensen for funksjonssymboler har hittil ikke vært mulig, fordi krav 7.1 forlangte $f_s \succsim_F f_{s'}$.

Det er ikke lett å vise terminering av systemet bestående av regelen $f(x:s') \longrightarrow f(g(x))$. Anta at vi klarte å vise $f_{s'}(x) \succ f_s(g_{s'}(x))$ for en termineringsordning $\succ$. La oss utvide regelsettet med en ny regel $(\forall x : s')\ \ g(x) \longrightarrow x$. Enhver forenklingsordning $\succ$ setter $g_{s'}(x) \succ x$ pga. subterm-kravet til forenklingsordninger. I så tilfelle ville vi hatt både $f_{s'}(x) \succ f_s(g_{s'}(x))$ og $g_{s'}(x) \succ x$, men systemet ville ikke terminere fordi det ville ha eksistert en uendelig omskrivningssekvens

$$f(x) \xrightarrow[\mathcal{R}]{} f(g(x)) \xrightarrow[\mathcal{R}]{} f(x) \xrightarrow[\mathcal{R}]{} \cdots$$

TT3-*mpo* vil klare å vise at systemet i eks. 8.1 terminerer, uten å "påvise" terminering av systemet utvidet med regelen $g(x) \longrightarrow x$, men det vil koste litt. TT3-metoden vil fremdeles være *statisk*, på den måten at gitt en presedens, så kan hver regel bedømmes for seg, slik at dersom vi har fastslått $v \succ h$, så gjelder det én gang for alle.

Et nytt eksempel forsterker også inntrykket av at nye vanskeligheter kan dukke opp dersom vi tillater $f_{s'} \succ_F f_s$ når $s' < s$.

Eksempel 8.2 La signaturen Σ og regelmengden $\mathcal{R}$ være

Typer:	s', s
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s$
	$a : s$
	$b : s'$
Regel:	$f(b) \longrightarrow f(a)$

Speilet blir regelen $f_{s'}(b) \longrightarrow f_s(a)$. Dersom man kan ha $f_{s'} \succ_F f_s$, så kan man ha også presedensen $f_{s'} \succ_F f_s \succ_F a \succ_F b$. Det vil gi $f_{s'}(b) \succ_{mpo} f_s(a)$ og alt er som ønsket. Denne presedensen ville selvfølgelig gi $a \succ_{mpo} b$, men regelsettet $\mathcal{R}$ utvidet med regelen $a \longrightarrow b$ gir et ikke-terminerende system, fordi vi har $f(b) \xrightarrow[\mathcal{R}]{} f(a) \xrightarrow[\mathcal{R}]{} f(b) \xrightarrow[\mathcal{R}]{} \cdots$.

Disse problemene har pga. typeminskningskravet til reglene ikke dukket opp tidligere.

En første tanke som slår en i eksemplene over, er at problemene oppstår fordi vi har $f_{s'} \succ_F f_s$ samtidig som det fins en regelspesialisering $l\mu \longrightarrow r\mu$ slik at $LS(l\mu) = s$ og $LS(r\mu) = s'$. Muligens skulle man forby denne typen regler? Det ville være altfor brutalt, som følgende eksempel viser:

Eksempel (forts. av eks. 8.1) Systemet i eks. 8.1 kan utvides med en regel $a{:}s \longrightarrow b{:}s'$ og vil, som jeg skal vise lenger ut i kapitlet, fremdeles være terminerende.

Det er heller ikke bare en regelspesialisering $l\mu \longrightarrow r\mu$ der $LS(l\mu) = s$ og $LS(r\mu) = s'$ som kan gi opphav til et F-sprang fra f_s til $f_{s'}$.

Eksempel 8.3 I det terminerende systemet $\mathcal{R}$ gitt ved

Typer:	s, s', s_1, s'_1
	$s' \leq s$
	$s'_1 \leq s_1$

Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s$
$g : s' \rightarrow s$
$h : s_1 \rightarrow s$
$h : s'_1 \rightarrow s'$
$a : s'$
$c : s_1$
$d : s'_1$
Regler: $(\forall x : s')\ f(x) \longrightarrow f(g(x))$
$c \longrightarrow d$

fins det ingen regelinstanser med typer hhv. s og s', men likevel kan det forekomme F-sprang fra f_s til $f_{s'}$ fordi $f(h(c)) \underset{\mathcal{R}}{\longrightarrow} f(h(d))$ er en omskrivning i $\mathcal{R}$ samtidig som $\lfloor f(h(c)) \rfloor = f_s(h_{s_1}(c))$ og $\lfloor f(h(d)) \rfloor = f_{s'}(h_{s'_1}(d))$.

For å være sikker på å unngå F-sprang fra f_w til $f_{w'}$ måtte muligens alle ekte typeminskende regler forbys, og det er et altfor strengt krav.

Det ligger altså mange fallgruber her, men siden eksempel 8.1 illustrerer en interessant grunn til at typeordnede omskrivningssystemer terminerer, går jeg videre med å definere en metode, kalt TT3-metoden, som tillater en (*mpo-*)presedens $f_{w'} \succ_F f_w$ når $w' < w$.

8.2 Nødvendigheten av TT3

En kan spørre seg hvor ofte det egentlig dukker opp omskrivningssystemer som man bare kan vise terminering av med en presedens $f_{w'} \succ_F f_w$[1]. Faktum er at alle regelspesialiseringer på formen $f_{w'}(\ldots) \longrightarrow f_w(\ldots)$ har samme minste type på både venstre og høyre side, slik at regelen hva gjelder typeminskning like gjerne kunne orienteres den andre veien, til $f_w(\ldots) \longrightarrow f_{w'}(\ldots)$. Dersom man for eksempel skal orientere en likning $f(x : s') \approx f(g(x))$ i signaturen i eks. 8.1 i en kompletteringsprosedyre, så kan denne orienteres $f(g(x : s')) \longrightarrow f(x)$.

Den første grunnen til at regler av denne typen kan være interessante er at man ikke bare bruker termineringsordninger til (Knuth-Bendix-) komplettering. I kompletteringsprosedyrer har vi en valgfrihet ved å kunne orientere en likninger i ønskelig retning. Mange programmer i de nyere (eksperimentelle) programmeringsspråkene er ikke annet enn omskrivningssystemer. I disse tilfellene er reglene gitt og kan selvfølgelig ikke "orienteres" den veien vi ønsker.

I tillegg kan også en kompletteringsprosedyre lage en regel som bare kan orienteres slik at for å påvise terminering må man ha en presedens $f_{s'} \succ_F f_s$ (der $s' < s$), noe som blir illustrert av følgende eksempel:

Eksempel 8.4 I systemet

Typer: s', s
$s' \leq s$

[1] Jeg antar som vanlig $w' < w$.

Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s$
$a : s$
$b : s'$
Likning: $(\forall x : s')\ \ f(x) \approx f(a)$

må likningen orienteres $f(x) \longrightarrow f(a)$ for å unngå å introdusere en ny variabel i en høyreside av en regel. For å vise $f_{s'}(x) \succ_{mpo} f_s(a)$ *må* vi ha $f_{s'} \succ_F f_s$.

I utypet omskrivning har teoretikerne, se f. eks. Plaisted [Pla93], vært forsiktige med å forlange eksplisitt at ingen regelhøyreside skal introdusere en ny variabel. Én av grunnene til at man ikke trenger å forlange dette eksplisitt, er at en variabelintroduksjon alltid ville føre til ikke-terminering i utypet omskrivning. Det er bare å instansiere den nye variabelen med samme regels venstreside! Dermed følger variabelkravet mer implisitt.

Systemet over viser at typeordnet omskrivning ikke nødvendigvis har denne egenskapen, fordi systemet med regelen $(\forall x : s')\ \ f(a) \longrightarrow f(x)$ faktisk ville terminere, da x ikke kan instansieres med f. eks. a eller med en term som har $f(a)$ som subterm. Det synes lite heldig å introdusere variable i reglenes høyresider siden man ikke vet hva variabelen skal instansieres med. Derfor mener jeg at man eksplisitt må forlange at ingen høyreside av en regel introduserer en ny variabel i typet omskrivning.

8.3 Idéen bak TT3

Det kan altså være nødvendig å ha en presedens $f_{w'} \succ_F f_w$ selv om $w' < w$. Tillater vi en slik presedens, mister vi imidlertid muligheten til å ha en ekstraregel

$$f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$$

i en speiling samtidig som f_w og $f_{w'}$ naturligvis heller ikke kan betraktes som "ekvivalente". Ekstrareglene ble innført for å håndtere F-sprang som oppsto ved bruk av en ekte typeminskende regel. Når ovenstående ekstraregel forsvinner, må man se nærmere på i hvilke tilfeller et F-sprang fra f_w til $f_{w'}$ kan forekomme. Et F-sprang kan forekomme i posisjon $p(1)$ som i følgende eksempel:

Eksempel 8.5 Omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ gitt ved

Typer: s', s
$s' \leq s$
Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s$
$g : s' \rightarrow s$
$a : s$
$b : s'$
Regler: $(\forall x : s')\ \ f(x) \longrightarrow f(g(x))$
$a \longrightarrow b$

terminerer, men det kan som kjent ikke vises med metoden TT2. Regelspeilingen blir $f_{s'}(x) \longrightarrow f_s(g_{s'}(x))$ og $a \longrightarrow b$. For å ha minste mulighet til å vise $f_{s'}(x) \succ_{mpo} f_s(g_{s'}(x))$ må vi ha $f_{s'} \succ_F f_s$. Dermed kan det ikke finnes en regel $f_s(x) \longrightarrow f_{s'}(x)$ i speilingen R'. Imidlertid fins en $\mathcal{R}$-omskrivning $f(a) \xrightarrow[\mathcal{R}]{} f(b)$, slik at en speiling $\langle F, \lfloor\ \rfloor, R''' \rangle$ må oppfylle $\lfloor f(a) \rfloor \xrightarrow[R''']{+} \lfloor f(b) \rfloor$, dvs. $f_s(a) \xrightarrow[R''']{+} f_{s'}(b)$. En løsning kunne være å definere en ny regel $f_s(a) \longrightarrow f_{s'}(b)$ i R'''.

En ekstraregel $f_s(x) \longrightarrow f_{s'}(x)$ kan altså erstattes dels av en regel

$$f_s(\lfloor l\mu \rfloor) \longrightarrow f_{s'}(\lfloor r\mu \rfloor)$$

for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ der $\lfloor f, LS(l\mu) \rfloor_F = f_s$ og $\lfloor f, LS(r\mu) \rfloor_F = f_{s'}$. Man kan legge merke til at denne regelen ikke svekker noen ordning; i og med at $f_s(\lfloor l\mu \rfloor) \xrightarrow[R''']{+} f_{s'}(\lfloor r\mu \rfloor)$ skal være en omskrivningssekvens i en speiling R''', må $f_s(\lfloor l\mu \rfloor) \succ f_{s'}(\lfloor r\mu \rfloor)$ gjelde for en termineringsordning $\succ$ vi ønsker å bruke.

Det er dessverre langt fra tilstrekkelig å erstatte ekstraregelen $f_s(x) \longrightarrow f_{s'}(x)$ med $f_s(\lfloor l\mu \rfloor) \longrightarrow f_{s'}(\lfloor r\mu \rfloor)$ da F-sprang som kjent kan forekomme andre steder enn i posisjon $p(1)$. Vi så i eks. 8.3 at omskrivningen $f(h(c)) \xrightarrow[\mathcal{R}]{} f(h(d))$ ga opphav til et F-sprang fra f_s til $f_{s'}$ i posisjon ϵ selv om systemet ikke inneholder noen ekte typeminskende regler f kan appliseres på.

Et F-sprang fra f_s til $f_{s'}$ kan altså også forårsakes av at et funksjonssymbol i Σ, symbolet h i eksempel 8.3, skifter verditype fra (f. eks.) s til s'. For å kunne representere et slikt F-sprang bør regelen

$$f_s(g_w(x_1, \ldots, x_n)) \longrightarrow f_{s'}(g_{w'}(x_1, \ldots, x_n))$$

være med i en speiling for alle deklarasjonspar $\langle g : w \rightarrow s_1,\ g : w' \rightarrow s_2 \rangle$ der $\lfloor f, s_1 \rfloor_F = f_s$ og $\lfloor f, s_2 \rfloor_F = f_{s'}$. Teorem 6.13 og lemmata 6.11 og 6.12 gir at $w' < w$ og $s_2 < s_1$ i dette tilfellet. For at ovenstående regel skal være $\succ_{mpo}$ -minskende må presedensen oppfylle $g_w \succ_F f_{s'}$ og $g_w \succ_F g_{w'}$. Vi kan imidlertid ha en omskrivningssekvens $f_s(g_w(l)) \xrightarrow[R''']{+} f_{s'}(g_{w'}(r))$ også når $g : w \rightarrow s$ og $g : w' \rightarrow s'$ ikke er deklarasjoner i Σ, men der $g_w, g_{w'}$ er funksjonssymboler i F. For å få testene på en enklere form, utvider jeg signaturen Σ på følgende måte: Dersom f_w er med i mengden F, som defineres på samme måte som tidligere, så utvider jeg Σ med en deklarasjon $f : w \rightarrow LS(\overline{x} : w)$. Denne utvidelsen er triviell og kan foretas helt automatisk.

Et første eksempel antyder hvordan en slik TT3-speiling kunne brukes:

Eksempel (forts. av eks. 8.1) Vi kan bruke ordningen *mpo* på systemet i eksempel 8.1 og kan vise at regelspeilingen $f_{s'}(x) \longrightarrow f_s(g_{s'}(x))$ terminerer med presedensen $f_{s'} \succ_F f_s \succ_F g_{s'}$. Det fins ingen regelspesialiseringer med en venstresides minstetype lik s og der tilsvarende høyreside har minstetype lik s'. Det fins heller ikke noen funksjonssymboler i signaturen med verditype lik både s og s', slik at ingen ekstratester er nødvendige. Dermed er $f_{s'}(x) \succ_{mpo} f_s(g_{s'}(x))$ og siden speilingen ikke trenger å inneholde andre regler har vi vist terminering av systemet.

Legg merke til at dersom man nå legger til en regel

$$(\forall x : s')\ \ g(x) \longrightarrow x$$

til regelmengden, så må man vise *både* $g_{s'}(x) \succ_{mpo} x$ *og* $f_s(g_{s'}(x)) \succ_{mpo} f_{s'}(x)$. Dette siste fordi det fins da en regelspesialisering, nemlig $g(x) \longrightarrow x$, der minstetypen til $g(x)$ er lik s, samtidig som minstetypen til $x : s'$ er lik s'. Fordi $f_{s'} \succ_F f_s$ må en speiling inneholde regelen $f_s(g_{s'}(x : s')) \longrightarrow f_{s'}(x)$. Det er umulig å vise $f_s(g_{s'}(x)) \succ_{mpo} f_{s'}(x)$, fordi $f_{s'}$ er større enn både f_s og $g_{s'}$ i $\succsim_F$. Dermed kan man ikke påvise terminering av (det utvidede) systemet (og det var bra i og med at det ikke terminerer!).

8.4 Idéen dessverre bare bra for unære funksjonssymboler

La meg nå forsøke bruke idéen til funksjonssymboler med aritet større enn 1.

Eksempel 8.6 La $\langle \Sigma, \mathcal{R} \rangle$ være følgende utvidelse av systemet i eksempel 8.5:

Typer:	s', s
	$s' \leq s$
Funksjoner:	$f : s, s \rightarrow s$
	$f : s, s' \rightarrow s$
	$g : s' \rightarrow s$
	$a : s$
	$b : s'$
Regler:	$(\forall y : s, x : s')\ \ f(y, x) \longrightarrow f(y, g(x))$
	$a \longrightarrow b$

Systemet terminerer opplagt, da det er samme system som i eksempel 8.5, med den forskjellen at f nå tar to argumenter. Den første argumentplassen (der y står) skjer det ingenting med. Speilet ned i det utypede planet blir regelen til $f_{s,s'}(y, x) \longrightarrow f_{s,s}(y, g_{s'}(x))$ og $f_{s,s'}(y, x) \succ_{mpo} f_{s,s}(y, g_{s'}(x))$ gjelder med presedensen $a \succ_F f_{s,s'} \succ_F f_{s,s} \succ_F g_{s'} \succ_F \ \succ_F b$. Det fins dessverre en omskrivning $f(y, a) \xrightarrow[\mathcal{R}]{} f(y, b)$ uten at $f_{s,s}(y, a) \succ_{mpo} f_{s,s'}(y, b)$. Fordi $f_{s,s'} \succ_F f_{s,s}$ måtte man ha enten $y \succ_{mpo} f_{s,s'}(y, b)$ eller $a \succ_{mpo} f_{s,s'}(y, b)$, men ingen av disse holder. Problemet gjelder for alle stiordninger.

For å omgå dette problemet må man enten forandre stiordningene, lage nye ordninger, eller finne andre eksisterende ordninger som takler dette problemet på en bedre måte.

8.4.1 Forandre stiordningene?

Det er allment akseptert at det er vrient å forbedre stiordningene. Hadde det vært mulig å gjøre enkle forbedringer, er de sannsynligvis gjort og publisert. Jeg har prøvd å styrke stiordningene på mange måter uten å lykkes.

8.4.2 En ny ordning?

En annen mulighet kunne være å definere en helt ny ordning som klarer problemene i TT3-idéen bedre. Jeg har ikke jobbet med dette. Den første grunnen til at jeg ikke har gjort noe her, er at jeg ville lage en generell metode for å vise terminering av typeordnet omskrivning.

Jeg vil at metoden skal gi brukbart resultat også for spesialtilfellet med bare én type, nemlig utypet omskrivning. Dersom jeg på kort tid spesialkonstruerte en ordning for TT3, ville den neppe være særlig sterk i det generelle tilfellet. Den andre grunnen er at man kan "gjenbruke" all teorien og implementasjon som finnes for disse vanligste ordningene.

For det tredje viser det seg at det faktisk fins ordninger som takler disse spesielle TT3-problemene veldig greit, og én av dem er klassikeren Knuth-Bendix-ordningen. I kapittel 10 skal jeg se på TT3-varianten av Knuth-Bendix-ordningen.

Den siste grunnen til at jeg ikke går videre med å prøve å finne en ny ordning er at jeg i kapittel 10 definerer en speiling som tillater at krav 7.1 også blir brutt av ikke-unære Σ-funksjonssymboler. Selv om teorien eksemplifiseres med ordningen *ukbo* som defineres i kapittel 9, vil selvfølgelig teorien, resultatene og fremgangsmåten der også gjelde enhver ordning som takler problemet.

8.4.3 Bare unære funksjonssymboler kan bryte krav 7.1

Enden på visen blir altså at bare unære funksjonssymboler kan bryte krav 7.1 når vi bruker *lpo* eller *mpo*.

8.5 TT3-metoden for stiordninger

TT3-*mpo*-metoden tar utgangspunkt i metoden TT2-*mpo*, der en ekstraregel $f_s(x) \longrightarrow f_{s'}(x)$ er erstattet med tidligere beskrevne regler $f_s(\lfloor l\mu \rfloor) \longrightarrow f_{s'}(\lfloor r\mu \rfloor)$ og $f_s(g_w(x_1, \ldots, x_n)) \longrightarrow f_{s'}(g_{w'}(x_1, \ldots, x_n))$ når $f_s \not\succsim_F f_{s'}$. Den første av disse er $\succ_{mpo}$ -minskende når $\lfloor l\mu \rfloor \succ_{mpo} f_{s'}(\lfloor r\mu \rfloor)$ mens den sistnevnte er $\succ_{mpo}$ -minskende når $g_w \succ_F g_{w'}$ og $g_w \succ_F f_{s'}$.

Metoden blir vesentlig enklere dersom alle for slappe deklarasjoner (se def. på s. 51) i den opprinnelige preregulære signaturen strammes inn, noe som er trivielt å utføre. Metoden TT3-*mpo* for å påvise terminering av typeordnede systemer blir da som beskrevet i figur 8.1 på side 114.

Bevis for korrekthet av TT3-*mpo*-metoden Beviset foretas i sin helhet i avsnitt 8.6.

8.5.1 Beregnbarhet av TT3-*mpo*-metoden

TT3-*mpo* må være beregnbar for at den skal brukes som prosedyre for å påvise terminering av et endelig omskrivningssystem i en *endelig* signatur. Det er lett å se at alle punktene i TT3-*mpo*-metoden er greit beregnbare i endelige systemer, siden et endelig system har endelig antall regelspesialiseringer, samtidig som $\lfloor\rfloor$ er enkel å beregne og det er algoritmisk avgjørbart hvorvidt et termpar er $\succ_{mpo}$ -minskende gitt en presedens $\succsim_F$.

Krav 4 til TT3-*mpo* gjør at metoden kan være forholdsvis tidkrevende, da man må teste *alle* par $\langle f_s, f_{s'} \rangle$ der $s' < s$ og $f_s \not\succsim_F f_{s'}$ mot *alle* deklarasjonspar $\langle g : w \rightarrow s_1 \, , \, g : w' \rightarrow s_2 \rangle$ der $w' < w$ og $\lfloor f, s_1 \rfloor_F = f_s$ og $\lfloor f, s_2 \rfloor_F = f_{s'}$. Det kan vises at denne testen generelt *ikke* har behagelige transitive egenskaper og virkelig må foretas for *alle* slike kombinasjoner $\langle f_s, f_{s'} \rangle$ og $\langle g : w \rightarrow s_1 \, , \, g : w' \rightarrow s_2 \rangle$.

Gitt: En preregulær typeordnet signatur Σ, der ingen type i Σ har uendelig mange typer under seg, og en mengde typeminskende regler $\mathcal{R}$ i $\mathcal{T}(\Sigma, X)$.

- Stram inn eventuelle for slappe deklarasjoner i Σ.
- La F, $\lfloor\ \rfloor$ og $\lfloor\ \rfloor_F$ være definert som i kapittel 6, og la $\succsim_F$ være en kvasiordning over F.
- Utvid signaturen Σ slik at

$$f_w \in F \Longrightarrow f \in \Sigma_{w \to LS(\overline{x}:w)}$$

  for alle funksjonssymboler f_w i F.
- Dersom kravene

  1. for alle ikke-unære funksjonssymboler g i Σ gjelder

$$w' < w \wedge g_{w'}, g_w \in F \Longrightarrow g_w \succsim_F g_{w'}$$

  2. for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$ gjelder

$$\lfloor l\mu \rfloor \succ_{mpo(\succsim_F)} \lfloor r\mu \rfloor$$

  3. for alle ekte typeminskende regelspesialiseringer $l\mu \longrightarrow r\mu$ der $\lfloor f, LS(l\mu) \rfloor_F = f_s$ og $\lfloor f, LS(r\mu) \rfloor_F = f_{s'}$ og $f_s \not\succsim_F f_{s'}$ gjelder

$$\lfloor l\mu \rfloor \succ_{mpo(\succsim_F)} f_{s'}(\lfloor r\mu \rfloor)$$

  4. for alle par $f_s, f_{s'}$ i F der $s' < s$ og $f_s \not\succsim_F f_{s'}$ gjelder

$$g_w \succ_F f_{s'} \quad \text{og} \quad g_w \succ_F g_{w'}$$

     for alle par av deklarasjoner $\langle g : w \to s_1,\ g : w' \to s_2 \rangle$ i (den utvidede) Σ når $w' < w \wedge s_2 < s_1$, $\lfloor f, s_1 \rfloor_F = f_s$ og $\lfloor f, s_2 \rfloor_F = f_{s'}$
  5. enten $\mathcal{R}$ eller Σ er endelig eller så er $\succsim_F$ velfundert

  er oppfylt så terminerer $\mathcal{R}$.

Figur 8.1: *Fremgangsmåten TT3-*mpo *for å vise terminering av typeordnede omskrivningssystemer.*

8.5.2 Eksempler på bruk av TT3-*mpo*

For å gjøre det enklere å følge beviset for at TT3-*mpo*-metoden er korrekt illustrerer jeg først hvordan TT3-mpo kan brukes for å vise terminering av noen av eksemplene i dette kapitlet. Eksemplenes kompleksitet gjør at jeg av leselighetsgrunner gjentar dem her.

Eksempel (forts. av eks. 8.2) Vi så tidligere at en presedens som tillater $f_{s'} \succ_F f_s$ kunne ha forårsaket problemer i følgende system:

Typer: s', s
$s' < s$
Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s$
$a : s$
$b : s'$
Regel: $f(b) \longrightarrow f(a)$

Terminering av denne kan vises ved $f_{s'}(b) \succ_{mpo} f_s(a)$ ved f. eks. $f_{s'} \succ_F f_s \succ_F a \succ_F b$ (eller selvfølgelig med mer opplagte $f_s \approx_F f_{s'}$ og $b \succ_F a$). Det fins ingen regelspesialiseringer $v \longrightarrow h$ i $\mathcal{R}$ der $LS(v) = s$ og $LS(h) = s'$ eller noen funksjonssymboler g med deklarasjoner $g : \ldots \rightarrow s$ og $g : \ldots \rightarrow s'$, slik at kravene 3 og 4 til TT3-*mpo* er trivielt tilfredsstilte.

Dersom man nå legger til en regel $a \longrightarrow b$ til $\mathcal{R}$, blir $\mathcal{R}$ ikke-terminerende, samtidig som $a \succ_{mpo} b$. Imidlertid inneholder (den utvidede) $\mathcal{R}$ en regel $a \longrightarrow b$ med $LS(a) = s$ og $LS(b) = s'$, som gjør at man pga. krav 3 også må vise $a \succ_{mpo} f_{s'}(b)$ for å ha TT3-*mpo*-terminering. Dette er umulig med presedensen over.

Eksempel (forts. av eks. 8.3) Systemet i eks. 8.3 var gitt:

Typer: $s, s', s_1 s'_1$
$s' \leq s$
$s'_1 \leq s_1$
Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s$
$g : s' \rightarrow s$
$h : s_1 \rightarrow s$
$h : s'_1 \rightarrow s'$
$a : a'$
$c : s_1$
$d : s'_1$
Regler: $(\forall x : s')\ \ f(x) \longrightarrow f(g(x))$
$c \longrightarrow d$

Reglene blir speilet $f_{s'}(x) \longrightarrow f_s(g_{s'}(x))$ og $c \longrightarrow d$ slik at for å vise at den første er $\succ_{mpo}$-minskende må $f_{s'} \succ_F f_s$ og $f_{s'} \succ_F g_{s'}$ gjelde. Siden $\langle h : s_1 \rightarrow s,\ h : s'_1 \rightarrow s' \rangle$ er et deklarasjonspar der $s'_1 < s_1$, $\lfloor f, s \rfloor_F = f_s$ og $\lfloor f, s' \rfloor_F = f_{s'}$, må vi pga. krav 4 til TT3-*mpo* ha $h_{s_1} \succ_F f_{s'}$ og $h_{s_1} \succ_F h_{s'_1}$. Presedensen

$$h_{s_1} \succ_F h_{s'_1} \succ_F f_{s'} \succ_F f_s \succ_F g_{s'} \succ_F c \succ_F d$$

oppfyller alle krav til TT3-*mpo* slik at vi har påvist at det endelige systemet $\mathcal{R}$ terminerer.

8.6 Korrekthetsbevis for TT3-*mpo*

Jeg skal i dette avsnittet bevise at $\mathcal{R}$ terminerer når punktene 1-5 i fig. 8.1 er oppfylt.

Jeg vil som tidligere definere en speiling $\langle F, \lfloor\ \rfloor, R''' \rangle$, deretter vise at $\langle F, \lfloor\ \rfloor, R''' \rangle$ virkelig er en speiling, og til sist vil jeg vise at R''' terminerer dersom krav 1-5 til TT3-*mpo* er oppfylt.

I likhet med beviset for korrekthet av TT2-metoden, vil to F-funksjonssymboler f_w og $f_{w'}$ være representert med samme element, nemlig ${f_w}^*$ i regelmengden R'''. Dersom det ikke har spesiell interesse, vil jeg ikke eksplisitt nevne disse mulighetene i beviset, slik at beviset vil ta utgangspunkt i termspeilingsfunksjonen $\lfloor\ \rfloor$ og ikke i $\lfloor\ \rfloor^*$. For å komme til det mest generelle tilfellet, må man da foreta samme "overgang" som fra TT1 til TT2. Denne har jeg allerede gjort én gang, og den er rettfrem.

I dette avsnittet antar jeg at Σ er en typeordnet signatur som ikke inneholder for slappe deklarasjoner og at $\mathcal{R}$ er en typeminskende regelmengde i $\mathcal{T}(\Sigma, X)$.

8.6.1 Speilingen $\langle F, \lfloor\ \rfloor, R''' \rangle$

Jeg bruker F og $\lfloor\ \rfloor$ som definert for TT1.

R''' vil bestå av følgende regelmengde:

1. I likhet med TT1 og TT2-speilingen inneholder R''' en regel

$$\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$$

   for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$.

2. Dersom $f_s \succ_F f_{s'}$ for et funksjonssymbol f i Σ, så er, som vanlig, regelen

$$f_s(x_1, \ldots, x_n) \longrightarrow f_{s'}(x_1, \ldots, x_n)$$

   med i R'''.

3. For alle deklarasjonspar $\langle g{:}w{\rightarrow}s, g{:}w'{\rightarrow}s' \rangle$ i Σ der $w' < w$ og $s' < s$, så er regelen

$$\lfloor f, s \rfloor_F(g_w(x_1, \ldots, x_n)) \longrightarrow \lfloor f, s' \rfloor_F(g_{w'}(x_1, \ldots, x_n))$$

   med i R''' for alle funksjonssymboler f i Σ der $\lfloor f, s \rfloor_F \not\succsim_F \lfloor f, s' \rfloor_F$.

4. For alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$ der $LS(l\mu) = s$ og $LS(r\mu) = s'$, og for alle funksjonssymboler f i Σ er regelen

$$\lfloor f, s \rfloor_F(\lfloor l\mu \rfloor) \longrightarrow \lfloor f, s' \rfloor_F(\lfloor r\mu \rfloor)$$

   med i R''' dersom $\lfloor f, s \rfloor_F \not\succsim_F \lfloor f, s' \rfloor_F$.

Grunnen til den tungvinte skrivemåten er at $\lfloor f, s \rfloor_F$ ikke nødvendigvis trenger å være lik f_s, fordi f_s kanskje ikke er med i F. For å lette leseligheten vil jeg i resten av kapitlet bruke notasjonen f_s også for $\lfloor f, s \rfloor_F$ når dette ikke vil føre til problemer eller misforståelser.

Det er muligens ikke innlysende at $\langle F, \lfloor\ \rfloor, R''' \rangle$ er en speiling, så la meg illustrere det med et eksempel.

Eksempel 8.7 Gitt systemet $\mathcal{R}$ ved

Typer: $s, s', s_1, s'_1, s_2, s'_2, s_3, s'_3, s_4$
$s' \leq s$
$s'_1 \leq s_1$
$s'_2 \leq s_2$
$s'_3 \leq s_3$

Funksjoner: $f : s_3 \rightarrow s_4$
$f : s'_3 \rightarrow s_4$
$g : s_2 \rightarrow s_3$
$g : s'_2 \rightarrow s'_3$
$g : s \rightarrow s_1$
$g : s' \rightarrow s'_1$
$h : s_1 \rightarrow s_2$
$h : s'_1 \rightarrow s'_2$
$l : s$
$r : s'$

Regler: $l \longrightarrow r$
$\vdots$

Det fins en avledning $f(g(h(g(l)))) \underset{\mathcal{R}}{\longrightarrow} f(g(h(g(r))))$, slik at speilingen må tilfredsstille

$$f_{s_3}(g_{s_2}(h_{s_1}(g_s(l)))) \overset{+}{\underset{R'''}{\longrightarrow}} f_{s'_3}(g_{s'_2}(h_{s'_1}(g_{s'}(r))))$$

Anta så at $f_{s'_3} \succ_F f_{s_3}$ og $g_{s'} \succ_F g_s$ bryter mot krav 7.1 for TT2. Observer at presedensen fører til $g_{s_2} \succ_F g_{s'_2}$ og $h_{s_1} \succ_F h_{s'_1}$ pga. krav 4 til TT3-*mpo*. Da er per definisjon reglene

$$f_{s_3}(g_{s_2}(x)) \longrightarrow f_{s'_3}(g_{s'_2}(x)) \quad (1)$$
$$h_{s_1}(x) \longrightarrow h_{s'_1}(x) \quad (2)$$
$$g_s(l) \longrightarrow g_{s'}(r) \quad (3)$$

med i R'''. Med bruk av hhv. reglene (1), (2) og (3) fins da en avledning

$$f_{s_3}(g_{s_2}(h_{s_1}(g_s(l)))) \underset{R'''}{\longrightarrow} f_{s'_3}(g_{s'_2}(h_{s_1}(g_s(l)))) \underset{R'''}{\longrightarrow} f_{s'_3}(g_{s'_2}(h_{s'_1}(g_s(l)))) \underset{R'''}{\longrightarrow} f_{s'_3}(g_{s'_2}(h_{s'_1}(g_{s'}(r))))$$

som ønsket.

8.6.2 Korrekthetsbeviset

Jeg skal nå vise at $\langle F, \lfloor\ \rfloor, R''' \rangle$ beskrevet over virkelig er en speiling av $\langle \Sigma, R \rangle$.

Lemma 8.1 *Anta at $t \xrightarrow[\mathcal{R}]{} t'$ er en typeordnet omskrivning med en regel brukt i posisjon p i t. For alle posisjoner $p(i)$, $0 \leq i \leq D_p$, i t gjelder*

$$t|_{p(i)} \xrightarrow[\mathcal{R}]{} t'|_{p(i)} \Longrightarrow \lfloor t \rfloor|_{p(i)} \xrightarrow[R''']{+} \lfloor t' \rfloor|_{p(i)}$$

Bevis: Beviset følger følgende induksjonsskjema:

$$\frac{Q(0); \qquad Q(1); \qquad \forall i,\ 2 \leq i \leq D_p \mid Q(i-2) \wedge Q(i-1) \Longrightarrow Q(i)}{\forall i,\ 0 \leq i \leq D_p \mid Q(i)}$$

der metasymbolet Q står for lemmaet over og parameteren til Q for i'en i lemmaet.

i=0 Skal vise $t|_p \xrightarrow[\mathcal{R}]{} t'|_p \Longrightarrow \lfloor t \rfloor|_p \xrightarrow[R''']{+} \lfloor t' \rfloor|_p$. Siden den typede omskrivningen skjedde i posisjon p, holder konklusjonen. For de intrikate detaljer, se tilsvarende bevis for TT1, teorem 6.15.

i=1 Anta at $t|_{p(1)}$ er på formen $f(\ldots, l\sigma, \ldots)$. Omskrivningen $p|_{p(1)} \xrightarrow[\mathcal{R}]{} t'|_{p(1)}$ er i dette tilfellet $f(\ldots, l\sigma, \ldots) \xrightarrow[\mathcal{R}]{} f(\ldots, r\sigma, \ldots)$. Jeg setter $\lfloor f(\ldots, l\sigma, \ldots) \rfloor = f_s(\ldots, \lfloor l\sigma \rfloor, \ldots)$ og $\lfloor f(\ldots, r\sigma, \ldots) \rfloor = f_{s'}(\ldots, \lfloor r\sigma \rfloor, \ldots)$. Det fins (som i beviset for teorem 6.15) en utypet substitusjon σ' slik at det fins en spesialisering μ slik at $\lfloor l\sigma \rfloor = \lfloor l\mu \rfloor \sigma'$ og $\lfloor r\sigma \rfloor = \lfloor r\mu \rfloor \sigma'$. Teorem 6.13 gir at $s' \leq s$.
Det er tre muligheter for forholdet mellom f_s og $f_{s'}$. Enten er $f_s = f_{s'}$ (tilsvarende med $f_s \approx_F f_{s'}$), og fordi $\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$ er regel i R''', så er

$$\lfloor t \rfloor|_{p(1)} = f_s(\ldots, \lfloor l\mu \rfloor \sigma', \ldots) \xrightarrow[R''']{} f_s(\ldots, \lfloor r\mu \rfloor \sigma', \ldots) = \lfloor t' \rfloor|_{p(1)}$$

Dersom $f_s \succ_F f_{s'}$ kan termen $f_s(\ldots, \lfloor r\mu \rfloor \sigma', \ldots)$ ved hjelp av ekstraregel (som eksisterer siden $s' < s$) $f_s(x_1, \ldots, x_n) \longrightarrow f_{s'}(x_1, \ldots, x_n)$ omskrives til $f_{s'}(\ldots, \lfloor r\mu \rfloor \sigma', \ldots) = \lfloor t' \rfloor|_{p(1)}$.
Anta så $f_s \not\succ_F f_{s'}$, da er f unær, og oppbygning av R''' (punkt 4) gir da at $f_s(\lfloor l\mu \rfloor) \longrightarrow f_{s'}(\lfloor r\mu \rfloor)$ er en regel i R'''. Denne gir $(f_s(\lfloor l\mu \rfloor))\sigma' \xrightarrow[R''']{} (f_{s'}(\lfloor r\mu \rfloor))\sigma'$ for en R'''-omskrivning i posisjon $p(1)$ med substitusjon σ'. Da følger $(f_s(\lfloor l\mu \rfloor))\sigma' = f_s(\lfloor l\mu \rfloor \sigma')$ og $(f_{s'}(\lfloor r\mu \rfloor))\sigma' = f_{s'}(\lfloor r\mu \rfloor \sigma')$ fordi σ' er en homomorfi. Dermed har vi ønskede $f_s(\lfloor l\sigma \rfloor) \xrightarrow[R''']{+} f_{s'}(\lfloor r\sigma \rfloor)$.

i=n $\geq$ 2 Skal altså vise

$$t|_{p(n)} \xrightarrow[\mathcal{R}]{} t'|_{p(n)} \Longrightarrow \lfloor t \rfloor|_{p(n)} \xrightarrow[R']{+} \lfloor t' \rfloor|_{p(n)}$$

Jeg antar som vist i bevisskjemaet lemmaet for $n-1$ og $n-2$, dvs.

$$t|_{p(n-1)} \xrightarrow[\mathcal{R}]{} t'|_{p(n-1)} \Longrightarrow \lfloor t \rfloor|_{p(n-1)} \xrightarrow[R''']{+} \lfloor t' \rfloor|_{p(n-1)} \qquad \text{(IH1)}$$

og

$$t|_{p(n-2)} \xrightarrow[\mathcal{R}]{} t'|_{p(n-2)} \Longrightarrow \lfloor t \rfloor|_{p(n-2)} \xrightarrow[R''']{+} \lfloor t' \rfloor|_{p(n-2)} \qquad \text{(IH2)}$$

der konklusjonen i begge hypotesene kan antas siden $t|_{p(n)} \xrightarrow[\mathcal{R}]{} t'|_{p(n)}$ impliserer $t|_{p(n-1)} \xrightarrow[\mathcal{R}]{} t'|_{p(n-1)}$ og $t|_{p(n-2)} \xrightarrow[\mathcal{R}]{} t'|_{p(n-2)}$ når $n \geq 2$.
La $t|_{p(n)} = f(\ldots, g(\ldots, t|_{p(n-2)}, \ldots), \ldots)$. I og med at omskrivningen foregår i posisjon p er $t'|_{p(n)}$ lik $f(\ldots, g(\ldots, t'|_{p(n-2)}, \ldots), \ldots)$. Måten $\lfloor\ \rfloor$ er definert på, gjør at $\lfloor f(\ldots, g(\ldots, t|_{p(n-2)}, \ldots), \ldots) \rfloor = f_s(\ldots, g_w(\ldots, \lfloor t \rfloor|_{p(n-2)}, \ldots), \ldots)$ og

$\lfloor f(\ldots, g(\ldots, t'|_{p(n-2)}, \ldots), \ldots)\rfloor = f_{s'}(\ldots, g_{w'}(\ldots, \lfloor t\rfloor|_{p(n-2)}, \ldots), \ldots)$. Også denne gang gir teorem 6.13 at $s' \leq s$ og $w' \leq w$.

Det fins nå 3 muligheter for forholdet mellom f_s og $f_{s'}$. Dersom $f_s = f_{s'}$ gir IH1 ønsket konklusjon automatisk.

Anta $f_s \succ_F f_{s'}$. Da inneholder R''' en "ekstra-regel" $f_s(x_1, \ldots, x_n) \longrightarrow f_{s'}(x_1, \ldots, x_n)$. Bruk av denne først og deretter IH1 gir følgende avledningssekvens

$$f_s(\ldots, \lfloor t\rfloor|_{p(n-1)}, \ldots) \xrightarrow[R''']{} f_{s'}(\ldots, \lfloor t\rfloor|_{p(n-1)}, \ldots) \xrightarrow[R''']{+} f_{s'}(\ldots, \lfloor t'\rfloor|_{p(n-1)}, \ldots)$$

som var den ønskede $\lfloor t'\rfloor|_{p(n)}$.

Anta til sist $f_s \not\succsim_F f_{s'}$. Da er f unær. Siden Σ ikke inneholder for slappe deklarasjoner, gir proposisjon 3.13 at Σ er monoton og dermed er $s_2 < s_1$ for deklarasjonsparet $\langle g : w \rightarrow s_1,\ g : w' \rightarrow s_2\rangle$ og punkt 3 i definisjonen av R''' gir da at R''' inneholder en regel $f_s(g_w(x_1, \ldots, x_m)) \longrightarrow f_{s'}(g_{w'}(x_1, \ldots, x_m))$. Bruk av IH2 og denne regelen gir da en avledning

$$f_s(g_w(\ldots, \lfloor t\rfloor|_{p(n-2)}, \ldots)) \xrightarrow[R''']{+} f_s(g_w(\ldots, \lfloor t'\rfloor|_{p(n-2)}, \ldots)) \xrightarrow[R''']{} f_{s'}(g_{w'}(\ldots, \lfloor t\rfloor|_{p(n-2)}, \ldots))$$

som er den ønskede $\lfloor t\rfloor|_{p(n)} \xrightarrow[R''']{+} \lfloor t'\rfloor|_{p(n)}$.

□

Teorem 8.2 *$\langle F, \lfloor\ \rfloor, R'''\rangle$ beskrevet i dette avsnitt er en speiling av et typeordnet omskrivningssystem $\langle \Sigma, \mathcal{R}\rangle$.*

Bevis: For alle $t, t' \in \mathcal{T}(\Sigma, X)$ slik at $t \xrightarrow[\mathcal{R}]{} t'$, skal $\lfloor t\rfloor \xrightarrow[R''']{+} \lfloor t'\rfloor$. Dette faktum følger rett fra lemma 8.1 ved å sette inn D_p for i, dvs. $p(i) = p(D_p) = \epsilon$. □

Nå kommer resultatet jeg er på jakt etter.

Teorem 8.3 *Dersom kravene 1-5 i TT3*-mpo-*definisjonen i figur 8.1 er tilfredsstilt, så terminerer $\mathcal{R}$.*

Bevis: Jeg har bevist at speilingen $\langle F, \lfloor\ \rfloor, R'''\rangle$ virkelig er en speiling av $\langle \Sigma, \mathcal{R}\rangle$. Teorem 6.5 sier at det da holder å vise at det utypede omskrivningssystemet R''' terminerer. Ved å se på oppbygningen av R''' ser man at alle reglene i R''' er $\succ_{mpo}$ -minskende:

1. $\lfloor l\mu\rfloor \longrightarrow \lfloor r\mu\rfloor$ er en regel i R''' for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$. Krav 2 til TT3-*mpo* gir $\lfloor l\mu\rfloor \succ_{mpo} \lfloor r\mu\rfloor$ for alle slike regler.
2. Regelen $f_s(x_1, \ldots, x_n) \longrightarrow f_{s'}(x_1, \ldots, x_n)$ er med i R''' fordi $f_s \succ_F f_{s'}$. Dette gir rett frem $f_s(x_1, \ldots, x_n) \succ_{mpo} f_{s'}(x_1, \ldots, x_n)$.
3. Regelen $f_s(g_w(x_1, \ldots, x_m)) \longrightarrow f_{s'}(g_{w'}(x_1, \ldots, x_m))$ er med i R''' fordi Σ inneholder et deklarasjonspar $\langle g : w \rightarrow s_1,\ g : w' \rightarrow s_2\rangle$ der $w' < w$ og $s_2 < s_1$ og der $f_s = \lfloor f, s_1\rfloor_F \not\succsim_F \lfloor f, s_2\rfloor_F = f_{s'}$. Da følger $f_s(g_w(x_1, \ldots, x_m)) \succ_{mpo} f_{s'}(g_{w'}(x_1, \ldots, x_m))$ fra krav 4 til TT3-*mpo*, fordi $\succsim_F$ må tilfredsstille $g_w \succ_F f_{s'}$ og $g_w \succ_F g_{w'}$.
4. Til slutt er $f_s(\lfloor l\mu\rfloor) \longrightarrow f_{s'}(\lfloor r\mu\rfloor)$ med i R''' fordi $f_s = \lfloor f, LS(l\mu)\rfloor_F \not\succsim_F \lfloor f, LS(r\mu)\rfloor_F = f_{s'}$. Krav 3 til TT3-*mpo* gir $\lfloor l\mu\rfloor \succ_{mpo} f_{s'}(\lfloor r\mu\rfloor)$, som er det samme som $f_s(\lfloor l\mu\rfloor) \succ_{mpo} f_{s'}(\lfloor r\mu\rfloor)$ fordi $f_s \not\succsim_F f_{s'}$.

Dermed er $\xrightarrow[R'']{}$ inneholdt i $\succ_{mpo}$, og siden $\succ_{mpo}$ er en forenklingsordning, er $\succ_{mpo}$ velfundert når Σ (og dermed F) er endelig. Hvis $\mathcal{R}$ er endelig, kan man som i beviset for korrekthet av TT1-*mpo* vise at enhver R'''-omskrivningssekvens er bygget opp av et endelig antall F-symboler. Når $\succsim_F$ er velfundert sier teorem 2.34 at $\succ_{mpo}$ er velfundert. I disse tre tilfellene kan det ikke forekomme en uendelig R'''-omskrivningssekvens slik at R''' og derfor også $\mathcal{R}$ er terminerende. □

8.7 Terminering av omskrivningssystemer utvidet med retrakter

Retrakter er nødvendige for å kunne beregne verdien til semantisk meningsfylte men syntaktisk ikke velformede uttrykk som $top(pop(push(push(empty, 5), 8)))$. Jeg indikerte i avsnitt 4.4 hvordan retrakter kan brukes i omskrivningssystemer. Retraktdeklarasjoner og retraktregler kan legges til automatisk til et omskrivningssystem. Det var derfor nedslående at eksempel 6.4 i avsnitt 6.1 viste at en slik utvidelse av et omskrivningssystem ikke alltid er termineringsbevarende.

Desto mer kjærkommet er det at dersom man har påvist terminering av et omskrivningssystem med én av fremgangsmåtene TT1-$\succ$, TT2-$\succ$ eller TT3-*mpo*, der $\succ$ er en vilkårlig forenklingsordning, så terminerer også systemet utvidet med retrakter! For korthetens skyld nøyer jeg meg med følgende teorem:

Teorem 8.4 *Anta at vi med metoden TT3-*mpo *har vist at et typeordnet omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$ terminerer. Da terminerer også systemet utvidet med en mengde retraktdeklarasjoner $r_{s>s'} : s \longrightarrow s'$ (der $s' < s$) og retraktregler på formen*

$$\textbf{\textit{if}} < x \text{ er av type } s' > \textbf{\textit{then}}\ r_{s>s'}(x) \longrightarrow x$$

Bevis: Det er opplagt at dersom et system uten betingede regler terminerer, så terminerer også systemet der noen av reglene har betingelser heftet ved seg. Jeg velger derfor å vise terminering av systemet $\langle \Sigma', \mathcal{R}' \rangle$ der Σ' er den utvidede signaturen og $\mathcal{R}'$ er $\mathcal{R}$ utvidet med retraktreglene på formen $r_{s>s'}(x) \longrightarrow x$. Siden TT3-metoden forlanger typeminskende omskrivningssystemer må jeg benytte meg av kjennskapen at denne regelen er typeminskende, da $LS(r_{s>s'}(x\sigma)) = s'$ og $LS(x\sigma) \leq s'$ hver gang regelen kan brukes. Dessuten legger vi merke til at enhver retraktdeklarasjon $r_{s>s'} : s \longrightarrow s'$ gir opphav til ett distinkt funksjonssymbol i Σ' og til ett nytt symbol i den utypede funksjonssymbolmengden F. Funksjonen $\lfloor\ \rfloor_F$ utvides ikke uventet med $\lfloor r_{s>s'}, s'' \rfloor_F = r_{s>s'}$ for enhver type s''[2]. La den opprinnelige presedensen $\succsim_F$ utvides slik at $r_{s>s'} \succ_F f_w$ for ethvert retraktsymbol $r_{s>s'}$ og ikke-retraktsymbol f_w i F. Da må jeg vise at de 5 kravene til TT3-*mpo*-terminering på side 114 er oppfylt i det nye systemet når de er oppfylt i det opprinnelige.

Krav 1 Trivielt oppfylt da *ingen* $r_{s>s'}$ i Σ' gir opphav til to eller flere F-symboler.

Krav 2 De nye reglene er på formen $r_{s>s'}(x) \longrightarrow x$ og

$$\lfloor r_{s>s'}(x\mu) \rfloor = r_{s>s'}(x) \succ_{mpo} x = \lfloor x\mu \rfloor$$

[2] Jeg ser ingen grunn til å legge enda fler indekser til disse symbolene.

gjelder alltid. Siden $r_{s>s'}(x) \succ x$ gjelder i enhver forenklingsordning $\succ$, er dette grunnen til at TT2-$\succ$ kan vise terminering av det utvidede systemet for enhver forenklingsordning $\succ$ når TT2-$\succ$ viser terminering av det opprinnelige systemet.

Krav 3 En retraktregelspesialisering $r_{s>s'}(x\mu) \longrightarrow x\mu$ kan være ekte typeminskende da $x\mu$ kan være av en minste type $s'' < s'$ samtidig som $LS(r_{s>s'})$ alltid har minste type s'. Vi må derfor vise

$$\lfloor r_{s>s'}(x\mu) \rfloor = r_{s>s'}(x) \succ_{mpo} f_{w'}(x) = f_{w'}(\lfloor x\mu \rfloor)$$

for diverse funksjonssymboler $f_{w'}$ der det fins et annet F-symbol f_w slik at $w' < w$. Siden $r_{s>s'} \succ_F f_{w'}$ gjelder da $f_{w'}$ *ikke* kan være et retraktsymbol følger ønskede ulikhet.

Krav 4 Ethvert retraktfunksjonssymbol gir bare opphav til én deklarasjon og ett F-symbol og krav 4 er dermed trivielt oppfylt.

Krav 5 Anta først at den opprinnelige $\succsim_F$ var velfundert. Da er også den utvidede presedensen $\succsim_F$ velfundert f. eks. ved at alle retraktsymbolene er $\approx_F$-like. Dersom $\mathcal{R}$ eller Σ er endelig, så har vi vist tidligere at enhver omskrivningssekvens i speilingen er bygget opp av et endelig antall ulike F-symboler. Innføringen av retrakter forandrer ikke på dette, da de eneste retraktsymbolene i en omskrivningssekvens i speilingen er de som er med i termen der omskrivningssekvensen starter. Krav 5 holder "direkte" når antall retraktdeklarasjoner er begrenset oppad.

Alle kravene til TT3-*mpo*-terminering er derfor tilfredsstilt av det utvidede systemet, som følgelig terminerer, når de er tilfredsstilt av det opprinnelige. □

Det er lettere å vise at teoremet også gjelder for TT1-*mpo* og TT2-*mpo*.

Hvordan henger så resultatet sammen med det faktum et en retraktutvidelse ikke alltid er termineringsbevarende? De systemene som ikke er termineringsbevarende under retraktutvidelse er såpass "spesielle" at en forenklingsordning ikke kan vise at de terminerer. Meget kort sagt synes tilfellet analogt med at det terminerende omskrivningssystemet $\{f(f(x)) \longrightarrow f(g(f(x)))\}$ utvidet med en regel $g(x) \longrightarrow x$ ikke lenger er terminerende.

Merk imidlertid at teoremet bare gjelder når $\mathcal{R}$ terminerer relativt til en regelmengde X som inneholder minst én variabel av hver type. $\mathcal{R}$ kan godt terminere relativt til $\emptyset$, samtidig som det utvidede systemet ikke gjør det, dette fordi $r_{s>s'}(t)$ kan være en grunnterm av type s' selv om den opprinnelige signaturen ikke inneholder noen grunntermer av type s'.

8.8 Svakheter og mulige forbedringer av TT3-metoden

8.8.1 Prosessen kan itereres – TT4, TT5, ...

Idéen jeg brukte for å komme fra TT2 til TT3 gikk i korthet ut på å erstatte hver "upassende" regel

$$f_s(x) \longrightarrow f_{s'}(x)$$

ved å analysere hva som kunne stå for x'en. Disse ekstrareglene skulle sikre speiling ved F-funksjonssymbolsprang, så det hele endte opp i en analyse av når disse sprangene forekommer.

Jeg innførte regler

$$f_s(\lfloor l\mu \rfloor) \longrightarrow f_{s'}(\lfloor r\mu \rfloor)$$

i R''', når $\lfloor f, LS(l\mu) \rfloor_F = f_s$ og $\lfloor f, LS(r\mu) \rfloor_F = f_{s'}$, og $f_s \not\succsim_F f_{s'}$. Dette var en grei utvidelse siden den ikke svekket $\succ_{mpo}$, da $f_s(\lfloor l\mu \rfloor) \xrightarrow[R']{+} f_{s'}(\lfloor r\mu \rfloor)$ er en mulig avledningssekvens, slik at $f_s(\lfloor l\mu \rfloor) \succ_{mpo} f_{s'}(\lfloor r\mu \rfloor)$ uansett *må* gjelde.

I tillegg innførte jeg en regel

$$f_s(g_w(x)) \longrightarrow f_{s'}(g_{w'}(x))$$

for alle nødvendige $\langle g_w, g_{w'} \rangle$-par. Dersom denne regelen er "upassende" med hensyn til ønsket presedens $\succsim_F$ kan den imidlertid sløyfes, slik at

$$f_s(g_w(x)) \longrightarrow f_{s'}(g_{w'}(x))$$

erstattes med

$$f_s(g_w(\lfloor l\mu \rfloor)) \longrightarrow f_{s'}(g_{w'}(\lfloor r\mu \rfloor))$$

for alle nødvendige regelspesialiseringer $l\mu \longrightarrow r\mu$, og med

$$f_s(g_w(h_{s_1}(x))) \longrightarrow f_{s'}(g_{w'}(h_{s_2}(x)))$$

for alle nødvendige deklarasjonspar $\langle h : s_1 \rightarrow w \ , \ h : s_2 \rightarrow w' \rangle$. På denne måten får man en fremgangsmåte TT4, og prosessen kan itereres ved at regelen

$$f_s(g_w(h_{s_1}(x))) \longrightarrow f_{s'}(g_{w'}(h_{s_2}(x)))$$

sløyfes og erstattes med ..., som i sin tur fjernes og blir erstattet av ..., og så videre og så videre.

Det synes ganske opplagt at de forskjellige testene på $\succsim_F$ ville bli enda mer arbeidskrevende, så jeg skriver om denne idéen mer som en teoretisk variant, som kan være nyttig til å påvise terminering av kompliserte systemer, som eksempelvis følgende:

Eksempel 8.8 (bruk av TT4-*mpo*) La et typeordnet omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$ være gitt ved

Typer: s, s', s_1, s'_1, w, w'
$s' \leq s$
$s'_1 \leq s_1$
$w' \leq w$

Funksjoner: $f : s \rightarrow s$
$f : s' \rightarrow s$
$g : s_1 \rightarrow s$
$g : s'_1 \rightarrow s'$
$h : w \rightarrow s$
$h : w' \rightarrow s'$
$a : s_1$
$b : s$

$c : s'$
$d : s_1$
$e : s'_1$

Regler: $(\forall x : s'_1)\ f(g(x)) \longrightarrow f(g(a))$
$(\forall x : s')\ f(x) \longrightarrow f(b)$
$d \longrightarrow e$

Leseren vil sannsynligvis være enig i at det ikke er innlysende at $\mathcal{R}$ terminerer. Det er heller ikke lett å *bevise* at systemet faktisk gjør det, men TT4-*mpo* klarer biffen.

Alle regelspesialiseringer i $\mathcal{R}$ må på vanlig måte representeres i en speiling, som altså må inneholde reglene

$$f_{s'}(g_{s'_1}(x)) \longrightarrow f_s(g_{s_1}(a)) \qquad (1)$$
$$f_{s'}(x) \longrightarrow f_s(b) \qquad (2)$$
$$d \longrightarrow e \qquad (3)$$

For å vise $\succ_{mpo}$ -minskning av regel (2), må vi ha $f_{s'} \succ_F f_s$. Krav 4 til TT3-*mpo* gjør da at $g_{s_1} \succ_F f_{s'}$ og $g_{s_1} \succ_F g_{s'_1}$, fordi TT3-speilingen R''' inneholder regelen $f_s(g_{s_1}(x)) \longrightarrow f_{s'}(g_{s'_1}(x))$. Med dette kravet til $\succsim_F$ gjelder ikke

$$f_{s'}(g_{s'_1}(x)) \succ_{mpo} f_s(g_{s_1}(a))$$

Bruker man TT4-metoden, sløyfes regelen $f_s(g_{s_1}(x)) \longrightarrow f_{s'}(g_{s'_1}(x))$ til fordel for de to reglene

$$f_s(g_{s_1}(d)) \longrightarrow f_{s'}(g_{s'_1}(e)) \qquad (4)$$
$$f_s(g_{s_1}(h_w(x))) \longrightarrow f_{s'}(g_{s'_1}(h_{w'}(x))) \qquad (5)$$

Alle disse 5 reglene er $\succ_{mpo}$ -minskende med presedensen

$$d \succ_F h_w \succ_F f_{s'} \succ_F f_s \succ_F g_{s_1} \succ_F g_{s'_1} \succ_F h_{w'} \succ_F a \succ_F b \succ_F c \succ_F e$$

TT4-*mpo* har nå påvist terminering av systemet der TT3-metoden ikke kunne klare det.

8.8.2 En svakhet

En svakhet ved TT3 og TT4 osv. er at de ikke tillater $g_{s'} \succ_F g_s$ (der $s' < s$) når de tilhørende deklarasjonene er

$$g : s \rightarrow s \quad \text{og} \quad g : s' \rightarrow s'$$

Grunnen er at krav 4 gjør at man må vise

$$g_s(g_s(x)) \succ_{mpo} g_{s'}(g_{s'}(x))$$

som er umulig når $g_{s'} \succ_F g_s$. Uansett hvor mye prosessen itereres, må

$$g_s(\ldots g_s(g_s(x))\ldots) \succ_{mpo} g_{s'}(\ldots g_{s'}(g_{s'}(x))\ldots)$$

vises, slik at forholdet $g_{s'} \succ_F g_s$ er utelukket i TT4-*mpo*, TT5-*mpo* osv. Dette faktum byr på uoverstigelige hindringer når man vil vise terminering av følgende system:

Eksempel 8.9 La signaturen Σ og det terminerende systemet $\mathcal{R}$ være gitt

Typer:	s', s
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s$
	$g : s \rightarrow s$
	$g : s' \rightarrow s'$
	$a : s$
Regel:	$(\forall x : s')\ \ f(g(x)) \longrightarrow f(g(a))$

Regelspeilingen gir $f_{s'}(g_{s'}(x)) \longrightarrow f_s(g_s(a))$. Forholdet $g_{s'} \succ_F g_s$ er som vist utelukket, slik at den eneste muligheten for å påvise TT3-*mpo*-terminering er ved $f_{s'} \succ_F f_s$ og $f_{s'} \succ_F g_s$. Dersom $f_{s'} \succ_F f_s$ gir TT3-speilingen R''' en ekstraregel $f_s(g_s(x)) \longrightarrow f_{s'}(g_{s'}(x))$ som man da ikke kan vise er $\succ_{mpo}$ -minskende. Prøver vi TT4 og så videre, får man uansett en ekstraregel

$$f_s(g_s \ldots (g_s(g_s(x))) \ldots) \longrightarrow f_{s'}(g_{s'} \ldots (g_{s'}(g_{s'}(x))) \ldots)$$

Man kan ikke vise at både denne regelen og $f_{s'}(g_{s'}(x)) \longrightarrow f_s(g_s(a))$ er $\succ_{mpo}$ -minskende. Alle mine forsøk er ubrukelige på dette systemet.

I dette tilfellet ligger nøkkelen i å foreta en fullstendig analyse av regeltypene slik at man ser at et F-sprang fra g_s til $g_{s'}$ faktisk ikke kan forekomme fordi alle regelinstanser er typebevarende.

8.8.3 Mulige forbedringer

Faktumet over kunne man benyttet seg av på både enkle og mer intrikate måter.

Den enkleste er triviell. Dersom alle regelspesialiseringer $l\mu \longrightarrow r\mu$ er typebevarende, dvs. der $LS(l\mu) = LS(r\mu)$, så trenger man faktisk *ingen* restriksjon på $\succsim_F$. Alle regelapplikasjonene blir selvfølgelig typebevarende hvis alle regelspesialiseringer er typebevarende, slik at ingen F-funksjonssymbolsprang forekommer i en posisjon $p(i)$ der $i \geq 1$. I det tilfellet blir alle ekstrareglene $f_s(\ldots) \longrightarrow f_{s'}(\ldots)$ og alle dens erstatninger overflødige i en speiling R'.

Metoden TT1 (som da vil være lik TT2 og TT3) uten restriksjoner på $\succsim_F$ er absolutt en fin måte å vise terminering på, som bl. a. ville kunne vise terminering av eksempel 8.9. Det er egentlig ikke så mye mer å si om denne fremgangsmåten enn at den er bra når den kan brukes, men i de fleste tilfeller tror tror ikke det ville være altfor sannsynlig med en typeordnet regelmengde som er typebevarende for alle spesialiseringer.

Fra dette ene ytterpunktet kan man jobbe videre med å finne en mer nøyaktig speiling R'. For alle ekte typeminskende regelspesialiseringer kunne man sjekke hvilke funksjonssymboler som kan appliseres på disse, og herfra sjekke hvilke funksjonssymbolene som kan gi F-funksjonssymbolsprang i posisjon $p(2)$ ut fra applisering av reglene i posisjon p og så videre og så videre.

Fordelen med en slik fremgangsmåte er at den er sterkere enn TT3, ulempen er at den er komplisert. Metoden mister også det statiske aspektet jeg har villet beholde. Med dette mener jeg at når man har gitt en presedens $\succsim_F$, så skal hver regel kunne behandles for seg

etter hvert som den dukker opp, f. eks. under komplettering. Det er uheldig hvis man hadde vist $\succ_{mpo}$ -minskning av en masse regler med en presedens $\gtrsim_F$, og så genereres plutselig en ny regel, som stiller helt andre krav til $\gtrsim_F$. Mine opprinnelige forslag har denne statiske egenskapen.

Teoretisk sett er en slik analyse imidlertid interessant, fordi prosessen kanskje ender opp med å lage en perfekt syntaktisk speiling dersom den går tilstrekkelig lenge.

8.8.4 TT3 for andre ordninger

En annen svakhet med TT3-*mpo* var at en ekstraregel $f_s(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ bare kunne erstattes når f er unær. Eksempel 8.6 viste at det kan være nødvendig å kunne erstatte en slik ekstraregel også når f ikke er unær. Stiordningene egnet seg som vi så ikke til å vise terminering av slike *generelle* TT3-speilinger.

Mange slags termineringsordninger kan brukes til generelle TT3-speilinger. Jeg velger å eksemplifisere TT3 med en annen mye omtalt ordning, Knuth-Bendix-ordningen [KB70]. For at metoden TT3-*kbo* skal være brukbart sterk, må imidlertid Knuth-Bendix-ordningen "moderniseres". I neste kapittel vil jeg derfor definere en kvasiordning, som jeg velger å kalle *utvidet Knuth-Bendix-ordning, ukbo*. Deretter viser jeg hvordan denne *ukbo* kan brukes i TT1-, TT2- og TT3-speilingene. I beskrivelsen av TT3-*ukbo* vil jeg så definere en generell TT3-speiling.

Kapittel 9

Utvidet Knuth-Bendix ordning

Dette kapitlet presenterer en termineringsordning $\succsim_{ukbo}$ for utypet omskrivning. Ordningen er en enkel utvidelse av Knuth-Bendix-ordningen, *kbo*, definert i [KB70]. Grunnen til at jeg i det hele tatt omtaler Knuth-Bendix-ordningen er altså at den kan brukes til å vise terminering av en mer generell TT3-speiling. Anledningen til at Knuth-Bendix-ordningen må utvides til å bygge på en kvasiordning over funksjonssymboler er at den skal kunne brukes til TT2-metoden[1].

9.1 Knuth-Bendix-ordningen

Knuth-Bendix-ordningen bygger på en vektlegging av termer og på en rangering av *alle* funksjonssymbolene i en *total* ordning[2] $>_F$.

Alle funksjonssymboler og variable gis en ikke-negativ heltallig vekt, og vekten til en term t fås ved å legge sammen vektene til alle symbolene i t. Mer formelt fins det, gitt en signatur F og en variabel-mengde X en funksjon

$$we : F \cup X \rightarrow \mathcal{N}$$

og en funksjon

$$vekt : T(F, X) \rightarrow \mathcal{N}$$

slik at $vekt(x) = we(x)$ for en variabel x, og $vekt(f(t_1, \ldots, t_n)) = we(f) + vekt(t_1) + \cdots + vekt(t_n)$ for $n \geq 0$. Følgende restriksjoner må legges på vektene og rangeringen:

1. Enhver konstant må ha positiv vekt.
2. Et unært funksjonssymbol må ha positiv vekt dersom det *ikke* er det største funksjonssymbolet i ordningen $>_F$.
3. En variabel har vekt lik minste vekten til en konstant i F.

Begrunnelse for disse restriksjonene er gitt i [KB70].

[1] og dermed også til TT3-metoden som jo er en utvidelse av TT2-metoden!

[2] En total ordning $>$ over en mengde S er en binær, transitiv og ikke-refleksiv relasjon slik at $\forall s, s' \in S \mid s \neq s' \Longrightarrow (s > s' \vee s' > s)$.

Definisjon 9.1 *Funksjonen* $n : (F \cup X) \times T(F,X) \rightarrow \mathcal{N}$ *defineres slik at* $n(s,t)$ *gir antall forekomster av symbolet* s *i termen* t.

Knuth-Bendix-ordningen $\succ_{kbo}$ defineres i [KB70] som følger:

Definisjon 9.2 (Knuth-Bendix-ordningen [KB70]) *Knuth-Bendix-ordningen* $\succ_{kbo}$ *over en mengde* $T(F,X)$ *der* F *er en mengde (utypede) funksjonsymboler defineres rekursivt på følgende måte:*

kbo-1: $$\frac{vekt(t) > vekt(t') \qquad \forall x \in X \mid n(x,t) \geq n(x,t')}{t \succ_{kbo} t'}$$

kbo-2: $$\frac{vekt(t) = vekt(x) \qquad x \in X \qquad i \geq 1}{t = f^i(x) \succ_{kbo} x}$$

kbo-3: $$\frac{vekt(t) = vekt(t') \qquad \forall x \in X \mid n(x,t) = n(x,t') \qquad f >_F g}{t = f(t_1,\ldots,t_m) \succ_{kbo} g(t'_1,\ldots,t'_n) = t'}$$

kbo-4: $$\frac{vekt(t) = vekt(t') \qquad \forall x \in X \mid n(x,t) = n(x,t') \qquad \langle t_1,\ldots,t_n\rangle \succ_{kbo}^{lex} \langle t'_1,\ldots,t'_n\rangle}{t = f(t_1,\ldots,t_n) \succ_{kbo} f(t'_1,\ldots,t'_n) = t'}$$

Det lar seg vise at $\succ_{kbo}$ er en forenklingsordning over $T(F,X)$. Stiordninger er ofte enklere å jobbe med enn *kbo*, siden man i stiordninger bare trenger å definere en presedens $\succsim_F$ mellom funksjonssymboler, mens *kbo* *både* forlanger en *total* ordning $>_F$ over F og en vektlegging av hvert funksjonssymbol. Det er ofte ikke-trivielt å vektlegge funksjonssymbolene slik at hver regel er $\succ_{kbo}$ -minskende. Det er klart enklere å bare definere en presedens $\succsim_F$ slik at en regelmengde er $\succ_{mpo}$ -minskende, selv om også dette er vist å være NP-hardt [Der87]. Ordningen *kbo* har i motsetning til stiordningene den vakre egenskapen at det ikke involverer prøving og feiling med hensyn til hvilke inferensregler man skal bruke eller hvilke subtermer som skal multiset-sammenlignes for å avgjøre hvorvidt $t \succ_{kbo} t'$.

Følgende eksempler viser at *kbo* og de vanlige stiordningene ikke er sammenlignbare i styrke:

Eksempel 9.1 La $R = \{f(x) \longrightarrow g(a,a),\ g(b,x) \longrightarrow f(x)\}$ være et utypet omskrivningssystem. Ordningen *mpo* kan ikke vise terminering av systemet, fordi $f(x) \succ_{mpo} g(a,a)$ forlanger $f \succ_F g$, men for å vise $g(b,x) \succ_{mpo} f(x)$ må da enten $b \succ_{mpo} f(x)$ eller $x \succ_{mpo} f(x)$ gjelde, noe som ikke er tilfelle.

Med vilkårlig $>_F$ og vektene $we(b) = 1000, we(a) = we(g) = we(x) = 1$ og $we(f) = 10$ vises terminering av systemet ved hjelp av *kbo*.

Eksempel 9.2 Regelen $f(a,x) \longrightarrow g(x,x)$ kan vises *mpo*-terminerende med presedensen $f \succ_F g$, men kravet til variable gjør at det er umulig å vise $f(a,x) \succ_{kbo} g(x,x)$.

9.2 En modernisert Knuth-Bendix-ordning

I likhet med "gammel" *mpo* og *lpo* har også *kbo* problem med ekvivalens mellom funksjonssymbolene. For ethvert par av forskjellige funksjonssymboler f og g er $f >_F g$ eller $g >_F f$. Dette kan, som følgende eksempel viser, være uheldig.

Eksempel 9.3 La et (utypet) omskrivningssystem R være gitt ved

$$f(a) \longrightarrow g(b) \quad (1)$$
$$g(a) \longrightarrow f(b) \quad (2)$$
$$a \longrightarrow b \quad (3)$$
$$h(a,b,b) \longrightarrow h(b,a,a) \quad (4)$$

At R terminerer ser man lett ved *lpo* med en presedens $f \approx_F g \succ_F a \succ_F b$. Dersom vi erstatter alle g'ene med f'er får vi et i termineringshenseende svakere system, men som likevel kan vises å terminere i *kbo* med en presedens $a >_F b$.

Ordningen *kbo* kan imidlertid ikke vise terminering av systemet slik det står, siden regel 3 forlanger $vekt(a) \geq vekt(b)$ for å være $\succ_{kbo}$-minskende, samtidig som regel 4 forlanger $vekt(a) \leq vekt(b)$, noe som tilsammen gir $vekt(a) = vekt(b)$. Regel 1 og 2 medfører da $vekt(f) = vekt(g)$. Dermed veier venstresiden og høyresiden like mye i regel 1 og 2. Dersom $f >_F g$, så er ikke $g(a) \succ_{kbo} f(b)$ og dersom $g >_F f$, så gjelder ikke $f(a) \succ_{kbo} g(b)$. Hadde f og g vært "like" i ordningen $>_F$, kunne man vist $f(a) \succ_{kbo} g(b)$ og $g(a) \succ_{kbo} f(b)$ ved $f =_F g$ og $a >_F b$.

Eksempelet antyder at Knuth-Bendix-ordningen også må "moderniseres" til å gjelde kvasiordninger. Dette har Dershowitz gjort i følgende generelle definisjon som imidlertid bare omhandler grunntermer.

Definisjon 9.3 (Generalisering av *kbo* til en kvasiordning [Der87]) *La $\succsim_F$ være en kvasiordning over en mengde F av funksjonssymboler med fast aritet og $\succsim_T$ en kvasiordning over mengden $T(F,\emptyset)$ av grunntermer. Den* generelle Knuth-Bendix-ordningen $\succsim_{gkbo}$ *over $T(F,\emptyset)$ er definert rekursivt ved følgende inferensregler:*

gkbo-1:
$$\frac{t \succ_T t'}{t \succsim_{gkbo} t'}$$

gkbo-2:
$$\frac{f(t_1,\ldots,t_m) \approx_T g(t'_1,\ldots,t'_n) \qquad f \succ_F g}{f(t_1,\ldots,t_m) \succsim_{gkbo} g(t'_1,\ldots,t'_n)}$$

gkbo-3:
$$\frac{f(t_1,\ldots,t_m) \approx_T g(t'_1,\ldots,t'_n) \quad f \approx_F g \quad \langle t_1,\ldots,t_m\rangle \succsim^{lex}_{gkbo} \langle t'_1,\ldots,t'_n\rangle}{f(t_1,\ldots,t_m) \succsim_{gkbo} g(t'_1,\ldots,t'_n)}$$

der $\succsim^{lex}_{gkbo}$ er den lexikografiske ordningen indusert av $\succsim_{gkbo}$.

Teorem 9.4 ([Der87]) *Dersom $\succsim_T$ er en kvasiforenklingsordning over en mengde $T(F,\emptyset)$ av grunntermer der funksjonssymbolene har fast aritet, slik at $f(\ldots,t,\ldots) \approx_T t$ bare gjelder dersom f er unær og størst under kvasiordningen $\succsim_F$ (dvs. $g \in F \Longrightarrow f \succsim_F g$), så er $\succ_{gkbo}$ en forenklingsordning over $T(F,\emptyset)$.*

Definisjonen av $\succ_{gkbo}$ er imidlertid altfor generell til å kunne brukes til implementasjoner, fordi den ikke sier noe mer om hva $\succsim_T$ skal være. I tillegg må ordningen utvides til å også omfatte variable.

Jeg kunne i prinsippet klare meg uten en modernisert variant av *kbo*, men jeg velger likevel å forbedre *kbo* av to grunner. I tillegg til at TT2- og TT3-algoritmene for å vise typeordnet

terminering blir klart enklere, kan man like gjerne benytte en sterkere variant dersom *kbo* likevel kan/bør brukes i visse tilfeller. Min variant av *kbo* vil være en enkel utvidelse av den opprinnelige *kbo*.

9.3 Utvidet *kbo*

Jeg definerer nå termineringsordningen *ukbo* som er den utvidete varianten av Knuth-Bendix-ordningen. Denne ordningen bygger på samme vektlegging som *kbo*, mens den totale ordningen $>_F$ blir erstattet med en kvasiordning $\succsim_F$ over F.

På vanlig måte får hvert symbol s i $F \cup X$ tilordnet et ikke-negativt heltall $we(s)$ slik at

- alle konstanter får positiv vekt
- bare de $\succsim_F$-største funksjonssymbolene f (dvs. $\forall g \in F \,|\, f \succsim_F g$) kan både være unære og ha vekt 0
- en variabel har vekt lik minste vekten til en konstant i F.

Jeg kan dermed definere en ordning $\succsim_{ukbo}$ på følgende måte:

Definisjon 9.5 (Utvidet Knuth-Bendix-ordning) *La F være en mengde (utypede) funksjonssymboler. La videre $\succsim_F$ være en kvasiordning over F og we en vektleggingsfunksjon til $F \cup X$ som oppfyller kravene nevnt over. Den* utvidede Knuth-Bendix-ordningen $\succsim_{ukbo}$ *over $T(F,X)$ defineres rekursivt på følgende måte:*

ukbo-1:
$$\frac{vekt(t) > vekt(t') \qquad \forall x \in X \,|\, n(x,t) \geq n(x,t')}{t \succsim_{ukbo} t'}$$

ukbo-2:
$$\frac{vekt(t) = vekt(x) \qquad x \in \mathcal{V}(t) \cap X}{t \succsim_{ukbo} x}$$

ukbo-3:
$$\frac{vekt(t) = vekt(t') \qquad \forall x \in X \,|\, n(x,t) \geq n(x,t') \qquad f \succ_F g}{t = f(t_1,\ldots,t_m) \succsim_{ukbo} g(t'_1,\ldots,t'_n) = t'}$$

ukbo-4:
$$\frac{vekt(t) = vekt(t') \quad \forall x \in X \,|\, n(x,t) \geq n(x,t') \quad f \approx_F g \quad \langle t_1,\ldots,t_m\rangle \succsim^{lex}_{ukbo} \langle t'_1,\ldots,t'_n\rangle}{t = f(t_1,\ldots,t_m) \succsim_{ukbo} g(t'_1,\ldots,t'_n) = t'}$$

Utvidelsen fra *kbo* skulle være forholdsvis grei. Regel *ukbo-2* gjør at $x \succsim_{ukbo} x$, som er nødvendig for at $\succsim_{ukbo}$ skal være refleksiv, og at $f_1(f_2(\ldots(f_N(x))\ldots)) \succsim_{ukbo} x$ siden vektkravet gjør at f_i'ene må være unære og ha vekt 0. Legg også merke til at $\succsim_{ukbo}$ ikke forlanger $n(x,t) = n(x,t')$ i reglene *ukbo-3* og *ukbo-4*. Det er strengt tatt ikke nødvendig, men det kan vises at $\succsim_{ukbo}$ er en kvasiordning over $T(F,X)$.

Før vi går videre, må vi karakterisere relasjonen $\approx_{ukbo}$, fordi man må vise $l \succ_{ukbo} r$ for alle regler $l \longrightarrow r$ i en regelmengde for å vise terminering, mens $\succ_{ukbo}$ bare er definert indirekte (ved $\succ_{ukbo} = (\succsim_{ukbo} \setminus \approx_{ukbo})$).

9.3.1 *ukbo*-likhet

Jeg ville oppnå en form for ekvivalens mellom forskjellige funksjonssymboler i ordningen $\succ_{ukbo}$. Det viser seg, ikke uventet, at to termer er *ukbo*-like dersom den ene termen kan fås fra den andre ved å erstatte en forekomst av et funksjonssymbol f med en forekomst av et *ekvivalent* funksjonssymbol g (som godt kan være lik f), der f og g er ekvivalente hvis og bare hvis $we(f)$ er lik $we(g)$ og $f \approx_F g$. Legg merke til at dette ikke utelukker at to funksjonssymboler f og g er ekvivalente selv om de har forskjellig aritet, noe som kan være nyttig i $\succ_{ukbo}$. To termer som har f og g i "samme posisjon" vil imidlertid aldri være $\approx_{ukbo}$ -ekvivalente hvis f og g har ulik aritet.

Lemma 9.6 *$t \approx_{ukbo} t'$ gjelder hvis og bare hvis t' kan fås fra t ved å bytte ut forekomster av funksjonssymboler med ekvivalente funksjonssymboler, der f og g er ekvivalente hvis og bare hvis $we(f) = we(g)$ og $f \approx_F g$.*

Det vil si at $\approx_{ukbo}$ er den minste relasjonen som tilfredstiller

- *$x \approx_{ukbo} y$ hvis og bare hvis $x = y$ dersom x er en variabel*
- *$f(t_1, \ldots, t_m) \approx_{ukbo} g(t'_1, \ldots, t'_n)$ hvis og bare hvis f og g er ekvivalente, $m = n$ $(m \geq 0)$ og for alle $i \in 1..m$ så er $t_i \approx_{ukbo} t'_i$*

Bevis: Siden $t \approx_{ukbo} t'$ betyr $t \succsim_{ukbo} t'$ og $t' \succsim_{ukbo} t$ må $vekt(t) = vekt(t')$. Lemmaet kan vises med en kikk på oppbygningen av termen t.

t **variabel** Anta at t er en variabel x. For å vise $x \succsim_{ukbo} t'$ kan bare regel *ukbo-2* benyttes siden $vekt(x) \leq vekt(t')$ for alle termer t', og fra denne følger at t' må være lik x.

t **sammensatt term** Anta at t er på formen $f(t_1, \ldots, t_m)$ der $m \geq 0$. Fordi $\approx_{ukbo}$ er symmetrisk kan t' ikke være en variabel ved argumentasjonen over. Da er $t' = g(t'_1, \ldots, t'_n)$ og $n \geq 0$. For at både $t \succsim_{ukbo} t'$ og $t' \succsim_{ukbo} t$ skal gjelde når $vekt(t) = vekt(t')$ må $f \approx_F g$. For å ha $t \succsim_{ukbo} t'$ og $t' \succsim_{ukbo} t$ må vi da ved *ukbo-4* i begge tilfeller av definisjonen ha

$$\langle t_1, \ldots, t_m \rangle \succsim^{lex}_{ukbo} \langle t'_1, \ldots, t'_n \rangle \qquad (1)$$
$$\langle t'_1, \ldots, t'_n \rangle \succsim^{lex}_{ukbo} \langle t_1, \ldots, t_m \rangle \qquad (2)$$

Dersom $m = n = 0$ er (1) og (2) trivielt oppfylte. Hvis $m \geq 1$ gir (1) $t_1 \succsim_{ukbo} t'_1$ (eller at t'_1 ikke eksisterer, men da holder i hvert fall ikke $t'_1 \succsim_{ukbo} t_1$) og (2) gir $t'_1 \succsim_{ukbo} t_1$ som tilsammen gir $t_1 \approx_{ukbo} t'_1$. Dermed forlanger (1) imidlertid $t_2 \succsim_{ukbo} t'_2$ samtidig som (2) forlanger $t'_2 \succsim_{ukbo} t_2$. Itereres denne tankegangen får vi $m = n$ og $t_i \approx_{ukbo} t'_i$ for alle i mellom 1 og n. $t_i \approx_{ukbo} t'_i$ gjør at alle disse har lik vekt, slik at vekten til f må være lik vekten til g.

□

Når $\approx_{ukbo}$ nå er kjent, kan *ukbo* brukes til å påvise terminering av utypede omskrivningssystemer.

Teorem 9.7 *La F være en mengde funksjonssymboler og R en mengde regler i $T(F,X)$. Dersom R eller F er endelige, eller dersom termene i enhver omskrivningssekvens $t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots$ bare inneholder et endelig antall ulike funksjonssymboler, så terminerer R hvis*

$$l \succ_{ukbo} r$$

gjelder for alle regler $l \longrightarrow r$ i R.

Bevis: Bevises i følgende avsnitt. □

9.3.2 Korrekthetsbevis for *ukbo*-terminering

I dette avsnittet beviser jeg teorem 9.7. Jeg vil først vise at $\succ_{ukbo}$ er en forenklingsordning over $T(F,\emptyset)$, og deretter vise $l \succ_{ukbo} r \Longrightarrow l\sigma \succ_{ukbo} r\sigma$ for alle grunnsubstitusjoner σ. Det kan dermed ikke finnes noen sekvens $t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots$ av grunntermer som er selvomsluttende.

For å vise at $\succ_{ukbo}$ over grunntermer er en forenklingsordning trenger jeg noen lemmaer og en ny kvasiordning $\geq_V$ over termer der $t \geq_V t'$ gjelder hvis og bare hvis $vekt(t) \geq vekt(t')$.

Lemma 9.8 *Relasjonen $\geq_V$ over termpar er en kvasiforenklingsordning.*

Bevis: Det må vises at $\geq_V$ er en monoton kvasiordning som i tillegg har subtermegenskapen. Siden $t \geq_V t'$ er det samme som $vekt(t) \geq vekt(t')$ gjelder opplagt alle egenskapene. □

Lemma 9.9 *$f(\ldots,t,\ldots) =_V t$ kan bare forekomme når f er unær og størst i $\succsim_F$.*

Bevis: Må altså vise at $vekt(f(\ldots,t,\ldots)) = vekt(t)$ kun kan være tilfelle når $f \succsim_F g$ for alle funksjonssymboler g i F, og at f er unær. I og med at enhver term har positiv vekt, *må* f være unær. For at $vekt(f(t)) = vekt(t)$ må f ha vekt 0, og da sier kravet til *we* at f må være størst i F. □

Teorem 9.10 *Relasjonen $\succ_{ukbo}$ er en forenklingsordning over grunntermer.*

Bevis: Dersom man erstatter $\succsim_T$ med $\geq_V$ i definisjon 9.3 av $\succsim_{gkbo}$, ser man at $\succsim_{gkbo}$ er det samme som $\succsim_{ukbo}$ over grunntermer. Siden jeg i lemma 9.8 har vist at $\geq_V$ er en kvasiforenklingsordning og i lemma 9.9 at $f(\ldots,t,\ldots) =_V t$ bare kan gjelde når f er unær og maksimal i $\succsim_F$, så gir teorem 9.4 at $\succ_{ukbo}$ er en forenklingsordning over $T(F,\emptyset)$. □

Ved teorem 2.8 er det da tilstrekkelig å vise

Teorem 9.11 *For alle termer t, t' i $T(F,X)$ gjelder*

$$t \succ_{ukbo} t' \Longrightarrow t\sigma \succ_{ukbo} t'\sigma$$

for alle grunnsubstitusjoner $\sigma : T(F,X) \rightarrow T(F,\emptyset)$.

Bevis: Jeg må (pga. at regel *ukbo-4* kombinerer $\succsim_{ukbo}$ og $\succ_{ukbo}$ i premisset) vise den sterkere varianten

$$(t \succsim_{ukbo} t' \Longrightarrow t\sigma \succsim_{ukbo} t'\sigma) \wedge (t \succ_{ukbo} t' \Longrightarrow t\sigma \succ_{ukbo} t'\sigma)$$

og denne vises[3] ved induksjon på oppbygningen av beviset for $t \succsim_{ukbo} t'$.

ukbo-1: $t \succ_{ukbo} t'$ (og dermed også $t \succsim_{ukbo} t'$) gjelder fordi $vekt(t) > vekt(t')$ og $\forall x \in X \mid n(x,t) \geq n(x,t')$. Da følger $vekt(t\sigma) > vekt(t'\sigma)$ og derfor $t\sigma \succ_{ukbo} t'\sigma$ trivielt av at en substitusjon substituerer en variabel med en term som har minst like stor vekt som variabelen. Se [Kir92] og [KB70] for detaljer.

ukbo-2: Da er enten $t = x = t'$ og $t\sigma \succsim_{ukbo} t'\sigma$ kan vises med regel *ukbo-4* (eller følger for den saks skyld av lemma 9.6).

Eller så er $t = f_1(f_2(\ldots(f_N(x))\ldots))$ og $t' = x$. Siden $\succ_{ukbo}$ over grunntermer er en forenklingsordning, innehar den subtermegenskapen og er transitiv, noe som betyr at $t\sigma = f_1(f_2(\ldots(f_N(x\sigma))\ldots)) \succ_{ukbo} f_2(\ldots(f_N(x\sigma))\ldots) \succ_{ukbo} \cdots \succ_{ukbo} f_N(x\sigma) \succ_{ukbo} x\sigma = t'\sigma$ der transitivitet av $\succ_{ukbo}$ gir $t\sigma \succ_{ukbo} t'\sigma$.

ukbo-3: Da er $vekt(t) = vekt(t')$, $\forall x \in X \mid n(x,t) \geq n(x,t')$ og $f \succ_F g$. Da er opplagt $vekt(t\sigma) \geq vekt(t'\sigma)$ og dersom $vekt(t\sigma) = vekt(t'\sigma)$ gir *ukbo-3* at $t\sigma = f(t_1\sigma,\ldots,t_m\sigma) \succsim_{ukbo} g(t'_1\sigma,\ldots,t'_n\sigma) = t'\sigma$ siden $f \succ_F g$. Etter lemma 9.6 er $t\sigma$ og $t'\sigma$ ikke $\approx_{ukbo}$-like og dermed gjelder $t\sigma \succ_{ukbo} t'\sigma$.

ukbo-4: I dette tilfellet er $vekt(t) = vekt(t')$ og $\forall x \in X \mid n(x,t) \geq n(x,t')$, noe som på vanlig måte gir $vekt(t\sigma) \geq vekt(t'\sigma)$ der $vekt(t\sigma) > vekt(t'\sigma)$ automatisk fører til $t\sigma \succ_{ukbo} t'\sigma$. La $t = f(t_1,\ldots,t_m)$ og $t' = g(t'_1,\ldots,t'_n)$, $f \approx_F g$ og $\langle t_1,\ldots,t_m\rangle \succsim^{lex}_{ukbo} \langle t'_1,\ldots,t'_n\rangle$.

Dersom $\langle t_1,\ldots,t_m\rangle \succsim^{lex}_{ukbo} \langle t'_1,\ldots,t'_n\rangle$ gjelder fordi $m > n$ og $\forall i \in 1..n \mid t_i \succsim_{ukbo} t'_i$ så gir induksjonshypotesene $t_i\sigma \succsim_{ukbo} t'_i\sigma$ og dermed $t\sigma \succsim_{ukbo} t'\sigma$ ved regel *ukbo-4*. Siden $t\sigma$ og $t'\sigma$ ikke kan være $\approx_{ukbo}$-like ved lemma 9.6 gjelder $t\sigma \succ_{ukbo} t'\sigma$.

Anta så at $\langle t_1,\ldots,t_m\rangle \succsim^{lex}_{ukbo} \langle t'_1,\ldots,t'_n\rangle$ fordi det fins en $j \in 1..m$ slik at $t_j \succ_{ukbo} t'_j \wedge (\forall k < j \mid t_k \succsim_{ukbo} t'_k)$. Da gir induksjonshypotesene at $t_j\sigma \succ_{ukbo} t'_j\sigma \wedge (\forall k < j \mid t_k\sigma \succsim_{ukbo} t'_k\sigma)$, noe som gjør at $t\sigma \succsim_{ukbo} t'\sigma$. Men når vi har $t_j\sigma \succ_{ukbo} t'_j$ så gjelder *ikke* $t' = g(t'_1\sigma,\ldots,t'_n\sigma) \succsim_{ukbo} f(t_1\sigma,\ldots,t_m\sigma) = t$. Dermed er $t\sigma \succ_{ukbo} t'\sigma$ i dette tilfellet.

Anta til sist at $\langle t_1,\ldots,t_m\rangle \succsim^{lex}_{ukbo} \langle t'_1,\ldots,t'_n\rangle$ kommer av at $m = n$ og $t_i \approx_{ukbo} t'_i$ for alle $i \in 1..m$. Da gjelder imidlertid *ikke* $t \succ_{ukbo} t'$ fordi $t' \succsim_{ukbo} t$ siden $\langle t'_1,\ldots,t'_n\rangle \succsim^{lex}_{ukbo} \langle t_1,\ldots,t_m\rangle$ slik at det er tilstrekkelig å vise $t\sigma \succsim_{ukbo} t'\sigma$ som følger av at induksjonshypotesene gir $\forall i \in 1..m \mid t_i\sigma \succsim_{ukbo} t'_i\sigma$.

□

Teorem 9.7 følger rett frem av teoremene 9.10 og 9.11.

Bevis for teorem 9.7: Hvis R eller F er endelig, så er termene i enhver sekvens $t_1 \xrightarrow[R]{} t_2 \xrightarrow[R]{} \cdots$ av grunntermer bygget opp av et endelig antall funksjonssymboler, slik at en uendelig sekvens av disse ville vært selvomsluttende. Teorem 9.11 gir $l \succ_{ukbo} r \Longrightarrow l\sigma \succ_{ukbo} r\sigma$ for alle regler $l \longrightarrow r$ i R og alle grunnsubstitusjoner σ, og da teorem 9.10 sier at $\succ_{ukbo}$

[3] Legg merke til at det f. eks. er tilstrekkelig å vise $t \succsim_{ukbo} t' \Longrightarrow t\sigma \succ_{ukbo} t'\sigma$ for å vise hele setningen.

er en forenklingsordning (og følgelig er monoton og transitiv) over grunntermer så gjelder $\underset{R}{\longrightarrow} \subseteq \succ_{ukbo}$ over $T(F, \emptyset)$. Siden $\succ_{ukbo}$ er en forenklingsordning over $T(F, \emptyset)$ fører dette også til at det ikke kan finnes selvomsluttende omskrivningssekvenser av grunntermer, og i vårt tilfelle kan det derfor heller ikke finnes en uendelig omskrivningssekvens. Siden R derfor terminerer for $T(F, \emptyset)$ sier lemma 2.5 at R terminerer for $T(F, X)$.

9.3.3 Et eksempel

La oss nå se hvordan $\succsim_{ukbo}$ kan brukes til å vise at systemet i eksempel 9.3 terminerer, noe som er umulig å vise med vanlig *kbo*.

Eksempel (forts. av eks. 9.3) Med vektleggingen $we(a) = we(b) = we(f) = we(g) = we(h) = 1$ og presedensen (som er den opplagte utvidelsen av) $a \succ_F b \succ_F h \succ_F f \approx_F g$ gjelder $l \succsim_{ukbo} r$ for alle de fire reglene i eks. 9.3. Samtidig er $l \not\approx_{ukbo} r$ slik at $l \succ_{ukbo} r$. Teorem 9.7 sier da at R terminerer.

Kapittel 10

TT-metodene basert på *ukbo*

Jeg skal i dette kapitlet vise hvordan den utvidede Knuth-Bendix-ordningen, *ukbo*, kan brukes til å vise terminering av typeordnet omskrivning og dette bidrar også til å illustrere hvor generelt anvendbar speilingsidéen er. Fremgangsmåten vil bli som for TT1 til TT3 for stiordninger definert i tidligere kapitler, noe som gjør at bevisene for det meste blir ganske like bevisene der, og at jeg i dette kapitlet bare skisserer flere av disse.

TT3-*ukbo* kommer til å ha den fordelen at $f_{w'} > f_w$ kan være en "presedens" også for ikke-unære funksjonssymboler f når $w' < w$. TT3-speilingen jeg definerer i dette kapittel blir derfor en mest mulig generell TT3-speiling. I likhet med TT3-*mpo* vil TT3-*ukbo*-metoden være ganske beregningskrevende.

I dette kapitlet vil funksjonssymbolmengden F, utvidelsen av den preregulære signaturen Σ, funksjonssymbolsmerkingsfunksjonen $\lfloor\ \rfloor_F$ og termspeilingsfunksjonen $\lfloor\ \rfloor$ være som de er beskrevet i kapittel 6. I tillegg vil Σ alltid være en preregulær typeordnet signatur der ingen type har uendelig mange typer mindre enn seg selv, mens $\mathcal{R}$ vil være en typeminskende regelmengde over $\mathcal{T}(\Sigma, X)$.

10.1 TT1-*ukbo*

Teoretisk bygger jeg på idéene jeg brukte for TT1-*mpo* med speilinger og "merkede" funksjonssymboler. Det vil si at for ethvert deklarasjonspar $\langle f : w \rightarrow s\ ,\ f : w' \rightarrow s' \rangle$ der $w' < w$, er $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ en ekstraregel i den speilende regelmengden R'. For at denne skal være $\succ_{ukbo}$ -minskende, må enten vekten til f_w være større enn vekten til $f_{w'}$, eller så må de ha samme vekt og $f_w \succ_F f_{w'}$ i presedensen $\succsim_F$. For enkelhetens skyld innfører jeg et nytt symbol for dette.

Definisjon 10.1 (*ukbo*-presedens for funksjonssymboler) *Gitt en vektleggingsfunksjon $we : F \cup X \rightarrow \mathcal{N}$ og en presedens $\succsim_F\ : F \times F \rightarrow Bool$ over en mengde F av (utypede) funksjonssymboler, definerer jeg relasjonen $\geq_{F_{ukbo}\ (we, \succsim_F)}$ over $F \times F$ ved at*

$$f \geq_{F_{ukbo}\ (we, \succsim_F)} g \iff (we(f) > we(g)) \vee (we(f) = we(g) \wedge f \succsim_F g)$$

Jeg skriver bare $\geq_{F_{ukbo}}$ for $\geq_{F_{ukbo}\ (we, \succsim_F)}$ når we og $\succsim_F$ fremgår av sammenhengen eller er uten betydelse.

Det er lett å se at $\geq_{F_{ukbo}}$ er en kvasiordning når $\succsim_F$ er en kvasiordning. Ekvivalensrelasjonen og den ikke-refleksive partielle ordningen indusert av $\geq_{F_{ukbo}}$ skriver jeg hhv. $\simeq_{F_{ukbo}}$ og $>_{F_{ukbo}}$. Litt grovt kan man si at $\geq_{F_{ukbo}}$ i TT-*ukbo*-metodene tilsvarer $\succsim_F$ i TT-*mpo*. Dermed kan TT1-*ukbo*-metoden for å påvise terminering av typeordnede systemer se ut som vist i figur 10.1.

- Definér den utypede funksjonssymbolmengden F som beskrevet i kapittel 6.
- Gitt en vektleggingsfunksjon $we : F \cup X \rightarrow Nat$ og en presedens $\succsim_F : F \times F \rightarrow Bool$ som tilfredsstiller kravene til *ukbo* og kravene
  1. $\forall f_w, f_{w'} \in F \mid w' < w \Longrightarrow f_w >_{F_{ukbo}} f_{w'}$
  2. for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$ så gjelder
  $$\lfloor l\mu \rfloor \succ_{ukbo} \lfloor r\mu \rfloor$$
  3. $\mathcal{R}$ eller Σ er endelig

  så terminerer $\mathcal{R}$.

Figur 10.1: *Metoden TT1-*ukbo *for å påvise terminering av et typeordnet omskrivningssystem* $\langle \Sigma, \mathcal{R} \rangle$.

Bevis for korrekthet av TT1-*ukbo*: Jeg må vise at $\mathcal{R}$ virkelig terminerer når jeg i figur 10.1 påstår at $\mathcal{R}$ gjør det. La $\langle F, \lfloor\ \rfloor, R' \rangle$ være speilingen i beviset for korrekthet av TT1-*mpo*-metoden. Krav 1 til TT1-*ukbo* gjør at alle ekstrareglene $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ er $\succ_{ukbo}$-minskende, mens krav 2 sier at alle opprinnelige regler i R' er $\succ_{ukbo}$-minskende. Dersom $\mathcal{R}$ eller Σ er endelig så er termene i enhver R'-omskrivningssekvens bygget opp av en endelig mengde F-funksjonssymboler. Etter teorem 9.7 terminerer R' og følgelig også $\mathcal{R}$ ved teorem 6.5. □

10.2 TT2-*ukbo*

Ordningen *ukbo* er definert med tanke på å ha mulighet for ekvivalens mellom funksjonssymboler og det utnytter jeg i TT2-*ukbo*. I likhet med TT2-*mpo* innebærer det at krav 1 til TT1-*ukbo*-terminering slakkes, slik at TT2-*ukbo*-metoden ser ut som vist i figur 10.2. Den eneste forskjellen fra TT1-*ukbo* er altså at $>_{F_{ukbo}}$ er erstattet av $\geq_{F_{ukbo}}$ i krav 1.

Bevis for korrekthet av TT2-*ukbo*-metoden: Må her bevise at $\mathcal{R}$ terminerer når kravene 1–3 til TT2-*ukbo*-metoden er oppfylt. La $\langle F, \lfloor\ \rfloor^*, R'' \rangle$ være speilingen for beviset for TT2-*mpo*-terminering brukte, dvs. der hvert funksjonssymbol f_w i TT1-speilingen var erstattet med den f-kanoniske representanten ${f_w}^*$. Den eneste forandringen er at kvasiordningen $\succsim_F$ i det beviset erstattes med kvasiordningen $\geq_{F_{ukbo}}$.

- Definér den utypede funksjonssymbolmengden F som tidligere.
- Gitt en vektleggingsfunksjon $we : F \cup X \rightarrow \mathcal{N}$ og en presedens $\succsim_F : F \times F \rightarrow Bool$ som tilfredsstiller kravene til *ukbo* og dersom kravene
  1. $\forall f_w, f_{w'} \in F \mid w' < w \Longrightarrow f_w \geq_{F_{ukbo}} f_{w'}$
  2. for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$ så gjelder
  $$\lfloor l\mu \rfloor \succ_{ukbo} \lfloor r\mu \rfloor$$
  3. $\mathcal{R}$ eller Σ er endelig

  er tilfredsstilt så terminerer $\mathcal{R}$.

Figur 10.2: *Metoden TT2*-ukbo *for å påvise terminering av et typeordnet system* $\langle \Sigma, \mathcal{R} \rangle$.

Ekvivalensklasser og f-kanoniske representanter defineres utifra ekvivalensrelasjonen $\simeq_{F_{ukbo}}$. Dette gir ingen strukturell forskjell fordi $\geq_{F_{ukbo}}$ og $\succsim_F$ begge er kvasiordninger med samme betingelser. R'' som fremkommer på denne måten er en speiling av $\mathcal{R}$. Da $\mathcal{R}$ eller Σ er endelig, er som tidligere vist termene i enhver R''-omskrivningssekvens bygget opp av endelig mange ulike F-symboler. Teorem 9.7 gir at det tilstrekkelig å vise at alle reglene i R'' er $\succ_{ukbo}$-minskende.

Alle de gjenværende "ekstrareglene" i R'' terminerer som følge av krav 1.

De "opprinnelige" reglene i R'' er på formen $\lfloor l\mu \rfloor^* \longrightarrow \lfloor r\mu \rfloor^*$. Termen $\lfloor l\mu \rfloor^*$ er lik $\lfloor l\mu \rfloor$ der ethvert funksjonssymbol f_w er erstattet med et $\simeq_{F_{ukbo}}$-ekvivalent funksjonssymbol ${f_w}^*$ med samme aritet som f_w. Lemma 9.6 sier da at for enhver term t er $\lfloor t \rfloor$ og $\lfloor t \rfloor^*$ ekvivalente i ordningen $\succsim_{ukbo}$ slik at vi har

$$\lfloor l\mu \rfloor \succ_{ukbo} \lfloor r\mu \rfloor \Longleftrightarrow \lfloor l\mu \rfloor^* \succ_{ukbo} \lfloor r\mu \rfloor^*$$

Krav 2 til TT2-ukbo-terminering forlanger nettopp $\lfloor l\mu \rfloor \succ_{ukbo} \lfloor r\mu \rfloor$ slik at alle "opprinnelige" regler i R'' er *ukbo*-minskende.

Alle reglene i R'' er altså *ukbo*-minskende og da terminerer $\mathcal{R}$. □

10.3 TT3-*ukbo*

Den egentlige grunnen til at jeg omhandler *ukbo* er at TT3-*mpo* har begrensningen at $f_{w'} \succ_F f_w$ bare kan forekomme når f er unær. Denne svakheten vil ikke TT3-*ukbo* lide av, selv om TT3-*ukbo*-metoden i likhet med TT3-*mpo* vil forlange mange tester.

Speilingen bygger på samme idé som speilingen for TT3-*mpo*. I prinsippet er det bare å erstatte funksjonssymbolpresedensen $\succsim_F$ der med $\geq_{F_{ukbo}}$. Jeg vil heller ikke her ta for meg tilfellene med $\simeq_{F_{ukbo}}$-likhet mellom F-symbolene. Dette behandles trivielt på samme måte

som overgangen fra TT1-*mpo* til TT2-*mpo*. For enkelhetens skyld vil i likhet med TT3-*mpo*-resonnementet også i TT3-*ukbo* anta at alle for slappe deklarasjoner i signaturene er strammet inn. Det er trivielt å stramme inn en preregulær signatur (ved at en for slapp deklarasjon $f : w \rightarrow s$ erstattes med en deklarasjon $f : w \rightarrow LS(f(\overline{x}:w))$) og etter proposisjon 3.13 oppnår vi den fordelen at Σ, og en eventuell utvidelse av denne, blir monoton.

Dersom $f_w \not\geq_{F_{ukbo}} f_{w'}$ (som alltid antatt $w' < w$) har en speiling R''' mistet ekstraregelen $f_w(x_1, \ldots x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$, og denne må erstattes på samme måte som for TT3-*mpo*, nemlig med en regel

$$f_w(x_1, \ldots, \lfloor l\mu \rfloor, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, \lfloor r\mu \rfloor, \ldots, x_n)$$

(der alle variablene $x_1, \ldots, x_n$ er forskjellige og ingen av dem forekommer i $\lfloor l\mu \rfloor$) og en regel

$$f_w(x_1, \ldots, g_{w_1}(y_1, \ldots, y_m), \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, g_{w_2}(y_1, \ldots, y_m), \ldots, x_n) \tag{10.1}$$

(der alle variablene $x_1, \ldots, x_n, y_1, \ldots, y_m$ er forskjellige) hver gang dette er nødvendig. Det kan være et F-sprang for hver ekte typeminskende regelspesialisering $l\mu \longrightarrow r\mu$ i $\mathcal{R}$ der $LS(l\mu) = s_i$ og $LS(r\mu) = s'_i$ og for hvert deklarasjonspar $\langle g : w_1 \rightarrow s_i \ , \ g : w_2 \rightarrow s'_i \rangle$ der $w_2 < w_1$ og $s'_i < s_i$ og det fins en sekvens av typer $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots s_n$ slik at

$$f_w = \lfloor f, s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_n \rfloor_F \not\geq_{F_{ukbo}} \lfloor f, s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n \rfloor_F = f_{w'}$$

For at ekstraregelen 10.1 skal være $\succ_{ukbo}$ -minskende, må vi forlange

$$[we(f_w) + we(g_{w_1}), f_w] \geq^{lex} [we(f_{w'}) + we(g_{w_2}), f_{w'}] \tag{10.2}$$

der $[n_1, f] \geq^{lex} [n_2, g]$ gjelder hvis og bare hvis enten $n_1 > n_2$ eller $n_1 = n_2$ og $f \succsim_F g$. Det er kanskje ikke innlysende at betingelse 10.2 viser $\succ_{ukbo}$ -minskning av ekstraregelen 10.1. Ekstraregelen forekommer bare i systemet når $f_s \not\geq_{F_{ukbo}} f_{s'}$. Hvis $f_w \not\geq_{F_{ukbo}} f_{w'}$ fordi $we(f_w) < we(f_{w'})$ så fører $we(g_{w_1}) + we(f_w) = we(g_{w_2}) + we(f_{w'})$ til at $we(g_{w_1}) > we(g_{w_2})$. Hvis nå $f_w \succ_F f_{w'}$ er regel 10.1 $\succ_{ukbo}$ -minskende ved *ukbo-3*, og dersom $f_w \approx_F f_{w'}$ er ekstraregelen $\succ_{ukbo}$ -minskende ved *ukbo-4* da $\langle x_1, \ldots, g_{w_1}(\ldots), \ldots, x_n \rangle \succsim^{lex}_{ukbo} \langle x_1, \ldots, g_{w_2}(\ldots), x_n \rangle$ siden $we(g_{w_1}) > we(g_{w_2})$ og venstre- og høyresiden i regel 10.1 aldri er $\approx_{ukbo}$ -like. I det andre tilfellet er $f_w \not\geq_{F_{ukbo}} f_{w'}$ fordi $we(f_w) = we(f_{w'})$ og $f_w \not\succsim_F f_{w'}$. Krav 10.2 impliserer i dette tilfellet $we(g_{w_1}) > we(g_{w_2})$ og $\succ_{ukbo}$ -minskning av ekstraregel 10.1 ved *ukbo-1*. Legg merke til at det er mulig å ha presedensen $f_w \succ_F f_{w'}$ selv om $f_{w'} >_{F_{ukbo}} f_w$.

Dette fører frem til metoden TT3-*ukbo* vist i figur 10.3 på side 139.

Legg merke til at det å vise $f_w(\lfloor l\mu \rfloor) \succ_{ukbo} f_{w'}(\lfloor r\mu \rfloor)$ i krav 2 er det samme som å vise

$$f_w(x_1, \ldots, x_{i-1}, \lfloor l\mu \rfloor, x_{i+1}, \ldots, x_n) \succ_{ukbo} f_{w'}(x_1, \ldots, x_{i-1}, \lfloor r\mu \rfloor, x_{i+1}, \ldots, x_n)$$

mens betingelse 3 altså sikrer $\succ_{ukbo}$ -minskning av ekstrareglene

$$f_w(x_1, \ldots, g_{w_1}(y_1, \ldots, y_m), \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, g_{w_2}(y_1, \ldots, y_m), \ldots, x_n)$$

10.3.1 Bevis av TT3-*ukbo*-terminering

Beviset blir nesten helt likt beviset for TT3-*mpo*-terminering på side 116. Jeg vil som alltid vise at regelmengden R''' er en speiling av $\mathcal{R}$, at enhver R'''-omskrivningssekvens er bygget opp av endelige mange F-symboler og at alle reglene i R''' er $\succ_{ukbo}$ -minskende. Teorem 9.7 gir da at R''' terminerer og teorem 6.5 sier at også $\mathcal{R}$ terminerer.

- Stram inn eventuelle for slappe signaturer i Σ.
- La F være definert på én av måtene beskrevet i kapittel 6.
- Utvid signaturen Σ slik at

$$f_w \in F \Longrightarrow f \in \Sigma_{w \to LS(\overline{x}:w)}$$

  gjelder for alle merkede symboler f_w i F.
- $\mathcal{R}$ terminerer hvis det fins en vektleggingsfunksjon $we : F \cup X \to \mathcal{N}$ og en presedens $\succsim_F$ over F slik at kravene

  1. For alle regler $l \longrightarrow r$ i $\mathcal{R}$ og for alle spesialiseringer μ av variable i l må

  $$\lfloor l\mu \rfloor \succ_{ukbo} \lfloor r\mu \rfloor$$

     være oppfylt.
  2. For alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$ som kan gi opphav til et F-funksjonssymbolsprang, det vil si dersom $LS(l\mu) = s_i$ og $LS(r\mu) = s'_i$, der $s'_i < s_i$, så må man kunne vise

  $$f_w(\lfloor l\mu \rfloor) \succ_{ukbo} f_{w'}(\lfloor r\mu \rfloor)$$

     for alle funksjonssymboler f i Σ dersom det fins en sekvens av typer $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$ slik at

  $$f_w = \lfloor f, s_1, \ldots, s_i, \ldots, s_n \rfloor_F \not\succsim_{F_{ukbo}} \lfloor f, s_1, \ldots, s'_i, \ldots, s_n \rfloor_F = f_{w'}$$

  3. For alle funksjonssymboler f i Σ og for alle deklarasjonspar $\langle g : w_1 \to s_i\,,\ g : w_2 \to s'_i \rangle$ der $w_2 < w_1$ og $s_i < s'_i$ og det fins en sekvens av typer $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$ slik at

  $$f_w = \lfloor f, s_1, \ldots, s_i, \ldots, s_n \rfloor_F \not\succsim_{F_{ukbo}} \lfloor f, s_1, \ldots, s'_i, \ldots, s_n \rfloor_F = f_{w'}$$

     må we og $\succsim_F$ oppfylle

  $$[we(f_w) + we(g_{w_1}), f_w] \geq^{lex} [we(f_{w'}) + we(g_{w_2}), f_{w'}]$$

  4. $\mathcal{R}$ eller Σ er endelig

  er tilfredsstilt.

Figur 10.3: *Fremgangsmåten TT3*-ukbo *for å vise terminering av et typeordnet omskrivningssystem* $\langle \Sigma, \mathcal{R} \rangle$.

Speilingen R'''

Regelmengden R''' består som nevnt av følgende 4 "deler":

1. $\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$ er med i R''' for alle regelspesialiseringer $l\mu \longrightarrow r\mu$ i $\mathcal{R}$.
2. På vanlig måte er ekstraregelen $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$ med i R''' når $f_w, f_{w'} \in F$ og $w' < w$ samtidig som $f_w >_{F_{ukbo}} f_{w'}$.
3. $f_w(x_1, \ldots, \lfloor l\mu \rfloor, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, \lfloor r\mu \rfloor, \ldots, x_n)$ er med i R''' når $w' < w$ og $l\mu \longrightarrow r\mu$ er ekte typeminskende slik at man ved bruk av en instans av denne regelspesialiseringen kan få et F-funksjonssymbolsprang fra f_w til $f_{w'}$ og der samtidig $f_w \not\geq_{F_{ukbo}} f_{w'}$.
4. For alle funksjonssymboler f i Σ er

   $$f_w(x_1, \ldots, g_{w_1}(y_1, \ldots, y_m), \ldots x_n) \longrightarrow f_{w_1}(x_1, \ldots, g_{w_2}(y_1, \ldots, y_m), \ldots x_n)$$

   med i R' dersom $f_w \not\geq_{F_{ukbo}} f_{w'}$, og $\langle g : w_1 \rightarrow s_i\ ,\ g : w_2 \rightarrow s'_i \rangle$ (der $w_2 < w_1 \wedge s'_i < s_i$) er et deklarasjonspar i Σ og det fins en sekvens av typer $s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots s_n$ slik at $\lfloor f, s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots s_n \rfloor_F = f_w$ og $\lfloor f, s_1, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots s_n \rfloor_F = f_{w'}$

Beviset for at R''' definert her virkelig er en speiling er veldig likt tilsvarende bevis for regelmengdespeilingen i TT3-*mpo*-beviset. Den eneste forskjellen er at noen unære funksjonssymboler er byttet ut med n-ære symboler.

Lemma 10.2 *Anta at $t \xrightarrow[\mathcal{R}]{} t'$ kommer av en (typeordnet) omskrivning med en regel brukt i posisjon p. For alle posisjoner $p(i)$, $0 \leq i \leq D_p$ gjelder*

$$t|_{p(i)} \xrightarrow[\mathcal{R}]{} t'|_{p(i)} \Longrightarrow \lfloor t \rfloor|_{p(i)} \xrightarrow[R''']{+} \lfloor t' \rfloor|_{p(i)}$$

Bevis: Beviset følger samme bevisskjema som beviset for lemma 8.1.

i=0 Skal vise $t|_p \xrightarrow[\mathcal{R}]{} t'|_p \Longrightarrow \lfloor t \rfloor|_p \xrightarrow[R']{+} \lfloor t' \rfloor|_p$ Siden den typede omskrivningen skjedde i posisjon p, holder konklusjonen.

i=1 Anta at $t|_{p(1)}$ er på formen $f(t_1, \ldots, l\sigma, \ldots, t_n)$. Beviset går som tilsvarende tilfelle i beviset for lemma 8.1. Jeg må vise at

$$f(t_1, \ldots, l\sigma, \ldots, t_n) \xrightarrow[\mathcal{R}]{} f(t_1, \ldots, r\sigma, \ldots, t_n)$$

fører til

$$\lfloor f(t_1, \ldots, l\sigma, \ldots, t_n) \rfloor \xrightarrow[R''']{+} \lfloor f(t_1, \ldots, r\sigma, \ldots, t_n) \rfloor$$

som er det samme som

$$f_w(\lfloor t_1 \rfloor, \ldots, \lfloor l\sigma \rfloor, \ldots, \lfloor t_n \rfloor) \xrightarrow[R''']{+} f_{w'}(\lfloor t_1 \rfloor, \ldots, \lfloor r\sigma \rfloor, \ldots, \lfloor t_n \rfloor)$$

Dersom $f_w >_{F_{ukbo}} f_{w'}$ gjøres dette som tidligere i to skritt ellers brukes den tilhørende regelen

$$f_w(x_1, \ldots, \lfloor l\mu \rfloor, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, \lfloor r\mu \rfloor, \ldots, x_n)$$

i R'''. Substitusjonen σ' blir substitusjonen σ' i beviset for lemma 8.1 utvidet med $\{x_1 \mapsto \lfloor t_1 \rfloor, \ \ldots\ , x_n \mapsto \lfloor t_n \rfloor\}$, som er grei utvidelse da ingen x_i er med i variabelmengden til $\lfloor l\mu \rfloor$. Resonnementet i tidligere nevnte bevis fører enkelt til konklusjonen.

i=n≥2 Skal altså vise

$$t|_{p(n)} \xrightarrow[\mathcal{R}]{} t'|_{p(n)} \Longrightarrow \lfloor t \rfloor|_{p(n)} \xrightarrow[R''']{+} \lfloor t' \rfloor|_{p(n)}$$

som følger av beviset for lemma 8.1. Eneste forskjellen er at $\succsim_F$ i dét beviset erstattes med $\geq_{F_{ukbo}}$ og at regelen

$$f_s(g_w(y_1, \ldots, y_n)) \longrightarrow f_{s'}(g_{w'}(y_1, \ldots, y_n))$$

erstattes med regelen

$$f_s(x_1, \ldots, g_w(y_1, \ldots, y_n), \ldots, x_m) \longrightarrow f_{s'}(x_1, \ldots, g_{w'}(y_1, \ldots, y_n), \ldots, x_m)$$

□

Herfra kan "TT2-steget" utføres slik at alle symboler f_w erstattes med den $\simeq_{F_{ukbo}}$ -like $f_w{}^*$, noe som ikke vil ha noe å si for bevisene siden $t \approx_{ukbo} t^*$. Vi kan dermed vise

Teorem 10.3 (Korrekthet av TT3-*ukbo*) *Dersom kravene 1-4 i TT3*-ukbo-*metoden i figur 10.3 er oppfylt så terminerer $\mathcal{R}$.*

Bevis: På samme måte som tidligere kan ekstrareglene forårsake at regelmengden R''' kan bestå av uendelige mange regler selv om $\mathcal{R}$ er endelig dersom Σ er uendelig. Imidlertid innfører ingen "ekstraregel" $f_w(x_1, \ldots, x_n) \longrightarrow f_{w'}(x_1, \ldots, x_n)$, $f_w(\ldots, g_{w_1}(\ldots), \ldots) \longrightarrow f_{w'}(\ldots, g_{w_2}(\ldots), \ldots)$ eller $f_w(\ldots, \lfloor l\mu \rfloor, \ldots) \longrightarrow f_{w'}(\ldots, \lfloor r\mu \rfloor, \ldots)$ nye Σ-symboler (som ikke forekommer i $\mathcal{R}$) i en omskrivningssekvens $t_1 \xrightarrow[R''']{} t_2 \xrightarrow[R''']{} \cdots$, slik at enhver slik sekvens er bygget opp av et endelig antall Σ-symboler (de i $\mathcal{R}$ og t_1). Da kan det (siden enhver type bare har endelige mange typer under seg, og siden typene går "nedover" disse ekstrareglene, f. eks. er $w' < w$ i tilfellene over) heller ikke forekomme et uendelig antall ulike F-symboler i $t_1, t_2, \ldots$. Det er lett å se at alle reglene i R''' er $\succ_{ukbo}$ -minskende når kravene 1-3 er tilfredsstilt, og når R''' er en speiling av $\mathcal{R}$ (lemma 10.2) der ingen R'''-omskrivningssekvens inneholder uendelige mange funksjonssymboler, så gir teoremene 6.5 og 9.7 at $\mathcal{R}$ terminerer. □

10.3.2 Beregnbarhet av TT3-*ukbo*

Det er betingelsene 2 og 3 i TT3-*ukbo*-metoden som er ekstra tidkrevende. Gitt en endelig signatur fins det bare et antall sekvenser av typer $s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n$ å undersøke i disse betingelsene. Endeligheten av signaturen, og dermed også av antall deklarasjoner gjør at intet funksjonssymbol i Σ har ubegrenset aritetslengde. Tallet n i sekvensen over er derfor begrenset oppad. Da $\succ_{ukbo}$ -minskning, $\lfloor\ \rfloor$ og $\lfloor\ \rfloor_F$ er enkelt beregnbare er det også beregnbart hvorvidt et endelig omskrivningssystem TT3-*ukbo*-terminerer gitt en presedens og en vektleggingsfunksjon.

10.3.3 Et eksempel

TT3-*ukbo* kan brukes til å vise at systemet i eksempel 8.6, som jeg av leselighetsgrunner gjentar her, terminerer. Dette er umulig å vise med noen av mine tidligere beskrevne fremgangsmåter.

Eksempel (forts. av eks. 8.6) Omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ var altså gitt ved

Typer:	s', s
	$s' \leq s$
Funksjoner:	$f : s, s \rightarrow s$
	$f : s, s' \rightarrow s$
	$g : s' \rightarrow s$
	$a : s$
	$b : s'$
Regler:	$(\forall y{:}s, x{:}s')\ \ f(y,x) \longrightarrow f(y, g(x))$
	$a \longrightarrow b$

Man må her i hvert fall vise $f_{s,s'}(y,x) \succ_{ukbo} f_{s,s}(y, g_{s'}(x))$ og $f_{s,s}(y,a) \succ_{ukbo} f_{s,s'}$ siden $f(y,a) \underset{\mathcal{R}}{\longrightarrow} f(y,b)$ er en omskrivning i $\mathcal{R}$. TT2-*ukbo* kan ikke brukes fordi den forlanger $f_{s,s} \geq_{F_{ukbo}} f_{s,s'}$.

Brukes TT3-*ukbo* er alle kravene til TT3-*ukbo*-terminering oppfylt med vektleggingen $we(a) = 1000$, $we(f_{s,s'}) = 199$, $we(b) = we(g_{s'}) = we(f_{s,s}) = 1$ og en vilkårlig presedens $\succsim_F$.

Også for TT-*ukbo*-metodene gjelder det at dersom man har påvist terminering av et system med TT1-*ukbo*, TT2-*ukbo* eller TT3-*ukbo*, så terminerer også systemet utvidet med retrakt-deklarasjoner og -regler. For TT3-*ukbo* kan dette vises på samme måte som teorem 8.4 ved å gi retraktsymbolene tilstrekkelig stor vekt. Til sist kan det nevnes at de alternative fremgangsmåtene for TT3-*mpo* som er foreslått i avsnitt 8.8.3 selvsagt også kan brukes for TT3-*ukbo*, fordi vi der restrikterer speilingen.

Kapittel 11

Gnaedigs metode og regeltransformasjoner

Dette kapitlet presenterer først metoden for å vise typeordnet terminering som er beskrevet i Gnaedig [Gn92a], den eneste registrerte[1] publikasjonen om typeordnet terminering. Jeg vil så vise at TT-metoden er ekte sterkere enn Gnaedigs metode. I forbindelse med denne ser jeg litt på forsøk på å transformere et omskrivningssystem til et annet (typeordnet) omskrivningssystem, som man med TT-metoden kan vise terminerer.

11.1 Gnaedigs metode

Metoden presentert i [Gn92a] tar utgangspunkt i regelen

$$is_even(y{:}NzNeg) \longrightarrow is_even(opp(y))$$

i eksempel 6.1 på s. 77. For å utnytte at y'en aldri kan instansieres med $opp(y)$ merker Gnaedig funksjonssymbolet is_even med aritetene $NzNeg$ og $NzPos$, slik at hun på denne måten får to (utypede) funksjonssymboler som kan brukes til å vise terminering med utypede teknikker. For å unngå det jeg har kalt F-sprang og problemer med substitusjon til en lavere type, forlanger metoden at signaturene skal være både regulære og *minimale*, der en signatur Σ er minimal hvis og bare hvis

$$f : w \rightarrow s \in \Sigma \ \wedge \ f : w' \rightarrow s' \in \Sigma \Longrightarrow (w = w' \vee w \bowtie w')$$

samtidig som regelmengdene skal være typeminskende. Gnaedigs metode blir beskrevet med min notasjon i figur 11.1 på neste side. Minimalitet sikrer at ingen F-sprang kan forekomme fordi typeminskning og monotoni-egenskaper tilsier at et F-sprang fra f_w til $f_{w'}$ bare kan forekomme når $w' < w$, og da $f_w, f_{w'}$ er med i F bare hvis $w = w'$ eller $w' \bowtie w$ fins det følgelig ingen F-sprang. Med denne metoden slipper man å *vise* $\lfloor l\mu \rfloor \succ_{mpo} \lfloor r\mu \rfloor$ eksplisitt for alle spesialiseringer μ, siden $\lfloor l \rfloor = \lfloor l\mu \rfloor$ for alle slike spesialiseringer.

TT-metoden danner et teoretisk grunnlag for å resonnere om terminering av typeminskende systemer. Dette skal jeg illustrere ved å bruke TT2-*mpo* til å påvise at Gnaedigs metode er korrekt.

[1] i CD-ROM-databasen og i hyllene på instituttbiblioteket

- La (den utypede) funksjonssymbolmengden F være gitt

$$F = \{f_w \mid f : w \rightarrow s \in \Sigma \text{ for én eller annen type } s\}$$

- Dersom

$$\lfloor l \rfloor \succ_{mpo} \lfloor r \rfloor$$

gjelder for alle regler $l \longrightarrow r$ i $\mathcal{R}$ og Σ eller $\mathcal{R}$ er endelig, så terminerer $\mathcal{R}$.

Figur 11.1: *Metoden presentert i [Gn92a] ("Gnaedigs metode") for å vise terminering av et typeminskende omskrivningssystem $\mathcal{R}$ i en regulær og minimal signatur Σ.*

Korrekthetsbevis av Gnaedigs metode: La hver deklarasjon i Σ gi opphav til ett symbol i F, noe som kan gjøres da Σ er antatt å være regulær. Siden $\lfloor t \rfloor = \lfloor t\mu \rfloor$ for alle spesialiseringer impliserer Gnaedig-kravet $\lfloor l \rfloor \succ_{mpo} \lfloor r \rfloor$ at TT2-*mpo*-kravet $\lfloor l\mu \rfloor \succ_{mpo} \lfloor r\mu \rfloor$ gjelder for alle regelspesialiseringer i $\mathcal{R}$. Kravet $w' < w \Longrightarrow f_w \succsim_F f_{w'}$ til TT2-*mpo* er trivielt oppfylt for alle presedenser $\succsim_F$ da $f_w, f_{w'} \in F \Longrightarrow w \bowtie w'$ når $w \neq w'$. Når $\mathcal{R}$ eller Σ er endelig er alle kravene til TT2-*mpo* tilfredsstilt og $\mathcal{R}$ terminerer. □

Det følger trivielt av resonnementet i beviset over at Gnaedigs metode er inneholdt i TT2-metoden i systemer med minimale signaturer, siden $\lfloor t \rfloor = \lfloor t\mu \rfloor$ samtidig som det ikke vil finnes F-symboler $f_w, f_{w'}$ slik at $w' < w$ som setter ekstra betingelser til $\succsim_F$. Det fins imidlertid mange omskrivningssystemer med minimale signaturer som Gnaedig-metoden ikke kan vise terminering av, men der TT2-metoden klarer dette. Et eksempel på dette er eks. 6.1 der signaturen blir minimal når de unødvendige deklarasjonene utelates fra signaturen, uten at Gnaedigs metode kan vise terminering av det nye systemet. TT-metoden er altså ekte sterkere enn Gnaedigs metode også i mengden av systemer med minimale signaturer. Dette er ikke spesielt overraskende i og med at jeg tok utgangspunkt i Gnaedigs idé og videreutviklet denne.

Ulempen med Gnaedigs metode er at den forlanger at signaturene skal være minimale. At dette er en kraftig begrensning illustreres bl. a. av at bare tre av de typeordnede signaturene i del II av denne oppgaven er minimale. Ikke engang signaturen i Gnaedigs eget eksempel 6.1 er minimal.

Konklusjonen må altså bli at man alltid kan bruke den mye mer generelle TT-metoden i stedet.

11.2 Regeltransformasjoner

Leseren stusser muligens over ovennevnte faktum at Gnaedigs metode *ikke* kan benyttes på eksemplet hun selv bruker for å illustrere sin metode. La oss derfor se på hva Gnaedig først gjør med systemet i eks. 6.1 for deretter å kunne bruke sin metode.

Signaturen Σ i eks. 6.1 er ikke minimal da $is_even : Int \rightarrow Bool$ og $is_even : NzPos \rightarrow Bool$ er to deklarasjoner i Σ samtidig som $NzPos < Int$ i typehierarkiet. Ut i fra en ikke beskrevet, men opplagt, analyse av at enhver grunnterm av typen Int også har type enten $NzPos$, $NzNeg$ eller $Zero$, fjerner hun deklarasjonen $is_even : Int \rightarrow Bool$ og erstatter den med deklarasjonen $is_even : Zero \rightarrow Bool$ slik at signaturen blir minimal. Denne prosessen med å fjerne og legge til deklarasjoner til en signatur for å få den minimal vil jeg kalle *minimalisering*. Nå er imidlertid høyresiden i regelen $(\forall x : Pos)\ is_even(s(s(x))) \longrightarrow is_even(x)$ ikke lenger en velformet term. Regelen erstattes derfor av de to reglene

$$\begin{aligned} (\forall x : NzPos)\ is_even(s(s(x))) &\longrightarrow is_even(x) \\ is_even(s(s(0))) &\longrightarrow is_even(0) \end{aligned}$$

Gnaedig kan nå bruke sin metode til å påvise at det *nye* systemet terminerer, selv om det er ikke gitt noen begrunnelse for at det opprinnelige systemet terminerer når det nye gjør det.

Minimalisering etterfulgt av regeltransformasjoner kan synes å være interessant, men virker dårlig på systemet

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$f : s' \rightarrow s'$
	$g : s \rightarrow s$
	$g : s' \rightarrow s'$
	$a : s$
Regler:	$g(a) \longrightarrow f(a)$
	$(\forall x : s')\ f(x) \longrightarrow g(x)$

som jeg i eksempel 6.13 har vist terminerer med TT2-*mpo*-presedensen $g_s \succ_F f_s \succsim_F f_{s'} \succ_F g_{s'}$. Signaturen er ikke minimal, men kan gjøres minimal ved f. eks. å fjerne deklarasjonene $f : s \rightarrow s$ og $g : s \rightarrow s$. Hva skjer da med regelen $g(a) \longrightarrow f(a)$? Hvis a får typen s' er det nye systemet ikke-terminerende. Dropper man istedet deklarasjonene $f : s' \rightarrow s'$ og $g : s' \rightarrow s'$ er det umulig å vise terminering av speilingen $\{g_s(a) \longrightarrow f_s(a),\ f_s(x) \longrightarrow g_s(x)\}$. En siste tanke kunne være å innføre en ny type s^* der $s' \bowtie s^*$ og erstatte deklarasjonene $f : s \rightarrow s$ og $g : s \rightarrow s$ med $f : s^* \rightarrow s$ og latt a hatt typen s^*. Et slikt forsøk på generalisering av minimaliseringsmetoden studerte jeg lenge, men resultatene ble langt fra bra. Jeg vil tro at Gnaedigs metode ikke kan utvides til å kunne vise at dette omskrivningssystemet terminerer.

Det er selvfølgelig ingen grunn til å se på regeltransformasjoner bare i sammenheng med minimalisering. Regelen $(\forall x : Pos)\ is_even(s(s(x))) \longrightarrow is_even(x)$ kunne godt erstattes med de tidligere nevnte reglene også uten at signaturen ble forandret. En regeltransformasjon etterfulgt av en TT2-*mpo*-analyse ville kunne vise at systemet i følgende eksempel terminerer:

Eksempel 11.1 La det typeordnede omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være

Typer:	$s, s_1, s_2, Bool$
	$s_1 \leq s$
	$s_2 \leq s$

Funksjoner: $f : s \rightarrow Bool$
$f : s_1 \rightarrow Bool$
$f : s_2 \rightarrow Bool$
$g : s \rightarrow s$
$g : s_1 \rightarrow s_1$
$g : s_2 \rightarrow s_1$
$opp : s \rightarrow s$
$opp : s_1 \rightarrow s_2$
$opp : s_2 \rightarrow s_1$
$b : s_1$

Regel: $(\forall x : s)\ f(g(x)) \longrightarrow f(opp(g(x)))$

$\mathcal{R}$ terminerer som vi skal se, men dette er umulig å vise med TT-*mpo*-metodene siden man da bl. a. måtte ha vist den ugyldige $f_s(g_s(x)) \succ_{mpo} f_s(opp_s(g_s(x)))$. Dersom regelen i $\mathcal{R}$ ble erstattet med reglene

$$(\forall x : s_1)\ f(g(x)) \longrightarrow f(opp(g(x)))$$
$$(\forall x : s_2)\ f(g(x)) \longrightarrow f(opp(g(x)))$$

vises terminering av det nye systemet siden både $f_{s_1}(g_{s_1}(x)) \succ_{mpo} f_{s_2}(opp_{s_1}(g_{s_1}(x)))$ og $f_{s_1}(g_{s_2}(x)) \succ_{mpo} f_{s_2}(opp_{s_1}(g_{s_2}(x)))$ gjelder med presedensen $f_{s_1} \succ_F f_{s_2} \succ_F opp_{s_1} \succ_F \cdots$. Jeg skal lenger ut i dette kapitlet vise at det opprinnelige systemet terminerer hvis og bare hvis det nye gjør det. Merk at man også kan vise terminering av det nye systemet med Gnaedigs metode etter at man i tillegg har fjernet deklarasjonene $f : s \rightarrow Bool$, $g : s \rightarrow s$ og $opp : s \rightarrow s$ fra signaturen.

Systemet har imidlertid mistet en "instans" $f(g(x : s) \longrightarrow f(opp(g(x)))$. Anta derfor at signaturen inneholder en konstant a av type s. Da gjelder $f(g(a)) \underset{\mathcal{R}}{\longrightarrow} f(opp(g(a)))$ i det opprinnelige systemet men ikke i det nye. Dersom det opprinnelige systemet også inneholdt en regel $opp(g(a)) \longrightarrow g(a)$ ville det *ikke* vært terminerende samtidig som man kunne "påvist" terminering ved å bruke nevnte regeltransformasjon (siden $opp(g(a)) \succ_{mpo} g(a)$ alltid gjelder). Merk også at typeminskningskravet gjør at en regel $(\forall x : s)\ opp(g(x)) \rightarrow g(x)$ ikke kan forekomme.

Det kan altså være endel å vinne på å foreta regeltransformasjoner ut i fra en analyse av variablene i reglene før man går løs på TT-metoden, men man må være uhyre forsiktig med å ikke forandre termineringsegenskaper. Jeg vil derfor ta en liten kikk på *hvordan* en regeltransformasjon ser ut og *når* den er ufarlig.

Definisjon 11.1 (Typeordnet regeltransformasjon) *Gitt en regulær typeordnet signatur* $\langle S, \leq, \Sigma \rangle$ *og en typeminskende regelmengde* $\mathcal{R}$ *består en* typeordnet regeltransformasjon *i at en regel*

$$(\forall \ldots, x : s, \ldots)\ l \longrightarrow r$$

i $\mathcal{R}$ *erstattes med* n *regler*

$$(\forall \ldots, x : s_1, \ldots)\ l \longrightarrow r$$
$$\vdots$$
$$(\forall \ldots, x : s_n, \ldots)\ l \longrightarrow r$$

der $n \geq 1 \wedge \forall i \in 1..n | s_i < s$ *samtidig som* $\forall s' \in S \mid s' < s \Longrightarrow (\exists j \in 1..n \mid s' \leq s_j)$.

Eksempler (forts. av eksemplene 6.1 og 11.1) I eksempel 11.1 ble altså regelen $(\forall x : s) \ldots$ byttet ut med $(\forall x : s_1) \ldots$ og $(\forall x : s_2) \ldots$, og i Gnaedigs tilfelle ble $(\forall x : Pos) \ldots$ erstattet av $(\forall x : NzPos) \ldots$ og $(\forall x : Zero) \ldots$ der $x : Zero$ kunne byttes ut med 0, den eneste ikke-variabel-termen av type $Zero$. Begge disse regeltransformasjonene oppfylte kravene til definisjonen over.

Eksempel 11.1 viste at en regeltransformasjon ikke alltid er ikke-termineringsbevarende, fordi vi ved en regeltransformasjon mister en omskrivning

$$t[l\sigma]_p \xrightarrow[\mathcal{R}]{} t[r\sigma]_p$$

når $\sigma = \{\ldots, x : s \mapsto t_i, \ldots\}$ der $LS(t_i) = s$. I prinsippet mister vi alltid en omskrivning siden $\sigma = \{\ldots, x : s \mapsto x : s, \ldots\}$ er en slik substitusjon. Følgende lemma sier imidlertid at dette siste ikke er noen hindring:

Lemma 11.2 *Anta at*

$$t_1 \xrightarrow[\mathcal{R}]{} t_2 \xrightarrow[\mathcal{R}]{} t_3 \xrightarrow[\mathcal{R}]{} \cdots$$

er en uendelig omskrivningssekvens i et omskrivningssystem $\mathcal{R}$. Da er også

$$t_1\mu \xrightarrow[\mathcal{R}]{} t_2\mu \xrightarrow[\mathcal{R}]{} t_3\mu \xrightarrow[\mathcal{R}]{} \cdots$$

er uendelig omskrivningssekvens i $\mathcal{R}$ når μ er en spesialisering av variable i t_1.

Bevis: Det er tilstrekkelig å vise

$$t \xrightarrow[\mathcal{R}]{} t' \Longrightarrow t\mu \xrightarrow[\mathcal{R}]{} t'\mu$$

Anta $t \xrightarrow[\mathcal{R}]{} t'$. Da fins det en regel $(\forall Y)\ l \longrightarrow r$ i $\mathcal{R}$, en substitusjon $\sigma : Y \rightarrow \mathcal{T}(\Sigma, X)$ og en posisjon p i t slik at $t|_p = l\sigma$ og $t' = t[r\sigma]_p$. La $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. Da er $\sigma' = \{x_1 \mapsto t_1\mu, \ldots, x_n \mapsto t_n\mu\}$ også en typeordnet substitusjon samtidig som $t\mu|_p = l\sigma'$ og $t'\mu = t\mu[r\sigma']_p$ siden μ er en homomorfi. Med substitusjonen σ' har vi den ønskede omskrivningen $t\mu \xrightarrow[\mathcal{R}]{} t'\mu$. □

Det gjør derfor ingenting om vi mister en omskrivning $t[l\sigma]_p \xrightarrow[\mathcal{R}]{} t[r\sigma]_p$ der $\sigma = \{\ldots, x \mapsto x, \ldots\}$. For ikke å miste noe vesentlig, må vi sørge for at det ikke finnes noen ikke-variabel-term $t\mu$ med minste type s. Følgende teorem sier i hvilke tilfeller det er sikkert å foreta en regeltransformasjon:

Teorem 11.3 *Anta at Σ er en regulær signatur, $\mathcal{R}$ en typeminskende regelmengde i Σ og X ev typeordnet variabelmengde som inneholder minst én variabel av hver type i signaturen. La $\mathcal{R}'$ være den typeminskende regelmengden man får når man foretar en typeordnet regeltransformasjon på en regel $(\forall \ldots, x : s, \ldots)\ l \longrightarrow r$ i $\mathcal{R}$ med hensyn på x. Anta videre at det fins en spesialisering μ av X slik for enhver term t i $\mathcal{T}(\Sigma, X)$ så er $LS(t\mu) \neq s$. Da terminerer $\mathcal{R}$ hvis og bare hvis $\mathcal{R}'$ terminerer.*

Bevis: "Bare hvis"-biten er triviell, og jeg ser derfor kun på "hvis"-delen. Ved lemma 11.2 er det tilstrekkelig å vise

$$t \xrightarrow[\mathcal{R}]{} t' \Longrightarrow t\mu \xrightarrow[\mathcal{R}']{} t'\mu$$

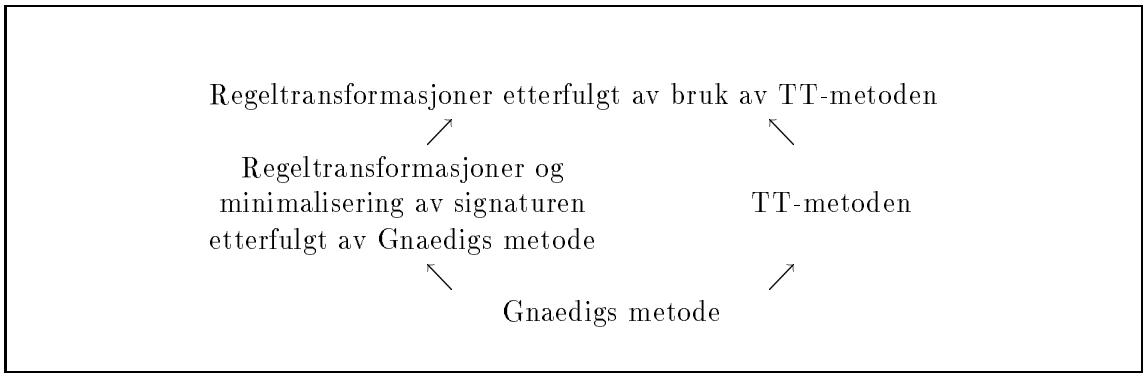
for alle termer $t, t' \in \mathcal{T}(\Sigma, X)$.

Anta at $t \xrightarrow[\mathcal{R}]{} t'$ er en omskrivning med bruk av en regel $(\forall x_1, \ldots, x_i : s, \ldots, x_n)\ l \longrightarrow r$ i posisjon p i t med en substitusjon $\sigma = \{x_1 \mapsto t_1, \ldots, x_i \mapsto t_i, \ldots, x_n \mapsto t_n\}$. Siden $LS(t_i) \leq s$ og $LS(t_i\mu)$ per antagelse er forskjellig fra s, er $LS(t_i\mu) = s' < s$. Dersom den brukte regelen ikke er transformert er det ingen problemer. Hvis regelen er transformert med hensyn på x_i har vi ved definisjonen av typeordnet regeltransformasjon likefullt en regel $(\forall x_1, \ldots, x_i : s'', \ldots, x_n)\ l \longrightarrow r$ i $\mathcal{R}'$ der $s' \leq s''$. Dermed har vi ønskede $t\mu \xrightarrow[\mathcal{R}']{} t'\mu$ ved bruk av ovennevnte regel med en substitusjon $\sigma = \{x_1 \mapsto t_1\mu, \ldots, x_i \mapsto t_i\mu, \ldots, x_n \mapsto t_n\mu\}$. □

Eksempler (forts. av eksemplene 6.11 og 11.1) Kravene over er tilfredstilt både i regeltransformasjonene i eksempel 6.1 og i eksempel 11.1, slik at disse regeltransformasjonene er ufarlige i termineringshenseende.

Merk at det aldri vil hjelpe mine metoder å foreta regeltransformasjoner den "andre veien", dvs. slå sammen reglene $(\forall \ldots, x : s_1, \ldots)\ l \longrightarrow r$, ... , $(\forall \ldots, x : s_n, \ldots)\ l \longrightarrow r$ til en regel $(\forall \ldots, x : s, \ldots)\ l \longrightarrow r$ der $s_i < s$ for alle $i \in 1..n$. Se kapittel 6 for begrunnelser for dette.

Hvis man likevel vil foreta regeltransformasjoner i et system, er det alltid minst like bra å bruke TT-metoden (dvs. TT2- og TT3-metodene) som Gnaedigs metode på det nye systemet, når vi lar F være gitt etter en fullstendig utvidelse av signaturen Σ og lar $f_{w'} \approx_F f_w$ når $w' < w$ og f_w er med i den minimale signaturens F-mengde. I lys av dette og eksemplene tidligere i kapitlet kan styrkeforholdet mellom de ulike metodene sammenfattes i figur 11.2 der **M**→**N** betyr at metode **N** er *ekte sterkere* enn metode **M**, dvs. at **N** kan vise terminering av alle omskrivningssystemer som **M** klarer å vise terminering av samtidig som det fins systemer som bare metode **N** kan vise terminerer.



Figur 11.2: *Skjematisk fremstilling av styrken til de forskjellige metodene for å vise typeordnet terminering.*

Vurdering av regeltransformasjoner

Eksempel 11.1 viste at det av og til er nødvendig å foreta regeltransformasjoner i det typeordnede systemet før man går løs på termineringsbevis med (f. eks.) TT-metoden.

Imidlertid er det meget sjelden det dukker opp systemer som man må transformere for å vise terminering.

Ulempene er at det synes å mangle teori om området når man ser bort i fra mitt lille teorem og at det veldig fort kan bli ganske komplisert å undersøke hvorvidt systemene er ekvivalente i termineringshenseende.

Kapittel 12

Vurdering og mulige forbedringer av TT-metoden

La meg nå vurdere TT-fremgangsmåten under ett og hvorfor ikke begynne med det hyggeligste, fordelene med metoden?

12.1 Fordeler

Enkelhet

Den største fordelen er at metoden (i hvert fall TT1 og TT2) er enkel, både å programmere men også bare for å avgjøre "for hånd" om et typeordnet system terminerer. Leseren kan sannsynligvis bekrefte dette ved å prøve å undersøke om systemene i denne oppgaven terminerer uten å benytte TT-metoden. Til enkelheten hører også at man slipper å gjøre en masse forandringer i (den regulære) signaturen og i regelmengdene.

Definerer en teori for typeordnet terminering

En annen stor fordel er at jeg med speilingene har definert et teoretisk grunnlag for typeordnet terminering. TT-metoden definerer bare speilinger og man kan bruke de kjente teknikkene for å vise at speilingene terminerer, eller man kan forbedre speilingene osv.

Forholdsvis sterk og generell

Vi har sett at TT-metoden er sterkere enn tidligere omtalte fremgangsmåter for å påvise typeordnet terminering, og merk at det å forlange $\lfloor l\mu \rfloor \succ \lfloor r\mu \rfloor$ *ikke* gjør fremgangsmåten svakere siden $\mathcal{R}$ og $\mathcal{R}' = \{l\mu \longrightarrow r\mu \mid l \longrightarrow r \in \mathcal{R} \wedge \mu$ spesialisering av variablene i $l\}$ kan sees på å være samme omskrivningssystem.

Fremgangsmåten har også den fordelen at den ikke stiller andre krav til systemene enn preregularitet og typeminskning.

Gjenbruk av teori for ordninger

Det har vært forsket mye rundt egenskaper, som f. eks. inkrementérbarhet, implementeringsegenskaper, velfunderthet osv., ved utypede termineringsordninger. All teorien som har fremkommet om dette kan brukes om igjen ved bruk av TT-metoden, slik at man ikke trenger å se på disse tingene enda en gang.

Greit beregnbar

Beregnbarhet (i endelige systemer) er et absolutt krav til en algoritme som skal påvise terminering av et system. Hver del i TT-fremgangsmåten er enkelt beregnbar i endelige systemer, og kompleksiteten (til TT1 og TT2) er normalt ikke mye større enn for å påvise terminering av utypede systemer. Man må også sette pris på den i kompletteringshenseende uunnværlige egenskapen at TT-metoden er statisk, som altså betyr at har man én gang vist $l \succ r$ så gjelder det én gang for alle.

Trenger ikke bekymre seg om retrakter

En stor fordel er at dersom TT1-$\succ$, TT2-$\succ$ (der $\succ$ er en vilkårlig forenklingsordning), TT3-*mpo*, TT3-*lpo* eller TT3-*ukbo* brukes til å vise terminering av et system, så terminerer også systemet utvidet med retrakter som derfor kan legges til automatisk.

TT-(*mpo*)-metoden kan automatiseres fullstendig

Gitt et typeordnet omskrivningssystem $\langle \Sigma, \mathcal{R} \rangle$, en utypet funksjonssymbolmengde F og en presedens $\succsim_F$ over F har vi sett at det er enkelt beregnbart hvorvidt betingelsene til TT1-*mpo*-, TT2-*mpo* og TT3-*mpo*-terminering er oppfylt. Det hadde imidlertid vært ønskelig om TT-metoden kunne automatiseres fullstendig, slik at en bruker av metoden slapp å tenke på å definere F og $\succsim_F$ selv. Kravet til en slik algoritme er at hver gang det *er mulig* å vise terminering av et system med TT-metoden, så skal algoritmen også gjøre dette.

Det viser seg at TT2-*mpo* og TT3-*mpo* kan automatiseres fullstendig. Definisjonen av F kan nemlig automatiseres siden jeg har vist at TT2-*mpo* og TT3-*mpo* gir best resultater når mengden F av merkede funksjonssymboler defineres $F = \{f_{w'} \mid \exists s | f \in \Sigma_{w \to s} \wedge w' < w\}$. Dermed er regelmengden $\{\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor\}$ automatisk beregnbar, slik at automatiseringen står og faller på i hvilken utstrekning en presedens $\succsim_F$ som viser terminering av systemet kan defineres automatisk.

Gitt en endelig signatur Σ er mengden F endelig, og dermed finnes det bare endelig mange kvasiordninger $\succsim_F$ over F. En algoritme kunne bare generere alle mulige presedenser over F og sjekke hvorvidt kravene til TT2-*mpo* eller TT3-*mpo* er oppfylt. TT-*mpo*-metoden forlanger altså ingen inngrep fra "brukeren".

En utfordring knyttet til en automatisering av TT-*mpo*-metoden ville være å gjøre den raskere, slik at algoritmen ikke genererer alle mulige presedenser over F. Jeg har ikke sett så mye på dette, men i jakten på en rask måte å definere presedenser, må vi huske på at det generelt er NP-hardt å finne en (eventuell) presedens $\succsim_F$ slik at $t \succ_{mpo(\succsim_F)} t'$ gjelder for to termer t og t' [Der87].

12.2 Ulemper

Jeg kan ikke komme bort fra at TT-metoden også har noen ulemper som må presenteres.

Bare for typeminskende systemer

TT-metoden forlanger, i likhet med Gnaedigs metode, typeminskende systemer. De fleste artiklene om typeordnet omskrivning og komplettering forutsetter typeminskende regelmengder. Under arbeid med feltet har jeg imidlertid følt veldig sterkt at typeminskning er et litt for strengt krav til typeordnet omskrivning. Jeg er derfor overbevist om at typeordnet omskrivning bare vil bli brukt til sitt fulle potensiale når man kan se bort fra dette kravet. Kravet om typeminskende systemer er derfor en reell ulempe ved TT-metoden, men jeg måtte ta utgangspunkt i eksisterende teori når jeg gikk løs på problemet.

Den senere tid har forskning i Frankrike (se f. eks. [HKK94]) forsøkt å utarbeide teori slik at typeminskningskravet til omskrivningssystemer kan fragås. Fleksibiliteten til TT-teorien viser seg ved at fremgangsmåten også kan modifiseres til å kunne brukes på ikke-typeminskende systemer. Det er bare å definere speilingen ut i fra de F-sprangene som kan oppstå, og forlange $f_{w_1} \succsim_F f_{w_2}$ hver gang et slikt sprang fra f_{w_1} til f_{w_2} kan forekomme. La meg illustrere denne (tenkte) fremgangsmåten med følgende eksempel:

Eksempel 12.1 La den regulære signaturen Σ og regelmengden $\mathcal{R}$ være gitt ved

Typer: s, s_1, s_2, s_3, s_4
$s_1 \leq s$
$s_2 \leq s$
$s_3 \leq s$
$s_4 \leq s$

Funksjoner: $f : s \rightarrow s$
$f : s_1 \rightarrow s$
$f : s_2 \rightarrow s$
$g : s \rightarrow s$
$g : s_3 \rightarrow s_1$
$g : s_4 \rightarrow s_2$
$h : s_1 \rightarrow s_2$
$a : s_3$
$b : s_4$

Regler: $a \longrightarrow b$
$(\forall x : s_2)\ f(x) \longrightarrow f(h(x))$

Det er forholdsvis lett å se at $\mathcal{R}$ terminerer, men dette kan ikke vises med noen hittil kjent metode, da regelen $a \longrightarrow b$ *ikke* er typeminskende. En enkel analyse av F-sprang gir at disse bare kan forekomme fra g_{s_3} til g_{s_4} siden $g(a) \xrightarrow[\mathcal{R}]{} g(b)$ er en omskrivning i $\mathcal{R}$ og fra f_{s_1} til f_{s_2} pga. omskrivningen $f(g(a)) \xrightarrow[\mathcal{R}]{} f(g(b))$. Den andre regelen er typebevarende og fører derfor ikke til F-sprang. Vi må derfor forlange $g_{s_3} \succsim_F g_{s_4}$ og $f_{s_1} \succsim_F f_{s_2}$ og samtidig vise

$$\begin{aligned} a &\succ_{mpo} b \\ \lfloor f(x : s_1) \rfloor = f_{s_1}(x) &\succ_{mpo} f_{s_2}(h_{s_1}(x)) = \lfloor f(h(x)) \rfloor \end{aligned}$$

for å påvise at $\mathcal{R}$ terminerer, noe som lar seg gjøre med presedensen $f_{s_1} \succ_F f_{s_2} \succ_F h_{s_1} \succ_F g_{s_3} \succsim_F g_{s_4} \succsim_F a \succ_F b$.

Den andre betingelsen i definisjonen av omskrivningsrelasjonen på side 60 introduserer flere ubehageligheter ved ikke-typeminskende systemer. Anta f. eks. at deklarasjonen $g : s \rightarrow s$ mangler i signaturen. Da ville $g(a) \xrightarrow[\mathcal{R}]{} g(b)$ ikke vært en omskrivning selv om begge termene er velformede, og *ingen* F-sprang kunne forekomme. Hvis deklarasjonen $g : s \rightarrow s$ er med i Σ samtidig som $f : s \rightarrow s$ mangler, ville $g(a) \xrightarrow[\mathcal{R}]{} g(b)$, men *ikke* $f(g(a)) \xrightarrow[\mathcal{R}]{} f(g(b))$ vært en lovlig omskrivning, slik at man bare måtte forlange $g_{s_3} \succsim_F g_{s_4}$ av presedensen.

Denne F-sprang-analysen kan utføres greit mekanisk, slik at TT-metoden enkelt kan modifiseres til å kunne brukes på ikke-typeminskende systemer.

Forlanger preregulære signaturer

Jeg har tidligere i oppgaven argumentert for at det normalt ikke gir mye mening å arbeide med signaturer der $\mathcal{T}(\Sigma, X)$ ikke er en fri typeordnet Σ-algebra over X. Regularitet sikrer at $\mathcal{T}(\Sigma, X)$ alltid er en slik fri algebra. Når TT-metoden attpåtil forlanger den enda mindre restriktive egenskapen preregularitet, skulle man tro at dette ikke utelukket mange slags systemer.

Waldmann [Wal92] viser som vi så på side 50 at det er tilstrekkelig å forlange at enhver grunnterm i $\mathcal{T}(\Sigma, \emptyset)$ bare har én tolkning i enhver Σ-algebra for at $\mathcal{T}(\Sigma, X)$ skal være en fri Σ-algebra over X. Den mest markante forskjellen er at konstanter kan ha ulike minste typer i slike "Waldmann-signaturer". Ved siden av det ville være litt mer problematisk å definere funksjonen $\lfloor\ \rfloor_F$, viser følgende eksempel at TT-fremgangsmåten ikke gir oss en speiling av et system $\langle \Sigma, \mathcal{R} \rangle$.

Eksempel 12.2 La en typeordnet signatur Σ som oppfyller Waldmanns krav til en typeordnet signatur og en regelmengde $\mathcal{R}$ være gitt ved

Typer:	s, s_1, s_2
	$s_1 \leq s$
	$s_2 \leq s$
Funksjoner:	$f : s, s \rightarrow s$
	$f : s_1, s_2 \rightarrow s$
	$g : s \rightarrow s$
	$g : s_1 \rightarrow s$
	$g : s_2 \rightarrow s$
	$a : s$
	$a : s_1$
	$a : s_2$
	$b : s$
Regler:	$(\forall x : s)\ f(x, x) \rightarrow b$
	$b \rightarrow f(a, a)$

Den første vanskeligheten er å beregne $\lfloor\ \rfloor$. Hva er f. eks. $\lfloor g(a) \rfloor$ lik? Problemet kunne løses ved sette $\lfloor g(a) \rfloor$ til å være *både* $g_{s_1}(a)$ og $g_{s_2}(a)$.

Definerer man en speiling for dette systemet på vanlig måte, får man regelmengden

$$\begin{array}{ll}
\text{for } x:s & \lfloor f(x,x)\rfloor = f_{s,s}(x,x) \longrightarrow b = \lfloor b\rfloor \\
\text{for } x:s_1 & \lfloor f(x,x)\rfloor = f_{s,s}(x,x) \longrightarrow b = \lfloor b\rfloor \\
\text{for } x:s_2 & \lfloor f(x,x)\rfloor = f_{s,s}(x,x) \longrightarrow b = \lfloor b\rfloor \\
 & \lfloor b\rfloor = b \longrightarrow f_{s_1,s_2}(a,a) = \lfloor f(a,a)\rfloor \\
\text{(ekstraregel)} & f_{s,s}(x,x) \longrightarrow f_{s_1,s_2}(x,x)
\end{array}$$

siden $\lfloor f(a,a)\rfloor$ ville være lik $f_{s_1,s_2}(a,a)$. Denne regelmengden R' ville *ikke* være en speiling av $\mathcal{R}$ siden $f(a,a)\xrightarrow[\mathcal{R}]{}b$ er en $\mathcal{R}$-omskrivning uten at vi har en tilsvarende omskrivningssekvens $\lfloor f(a,a)\rfloor = f_{s_1,s_2}(a,a)\xrightarrow[R']{+}b$. Dette er meget uheldig, siden man med presedensen $f_{s,s} \succ_F b \succ_F f_{s_1,s_2} \succ_F a$ kan vise terminering av R' med $\succ_{mpo}$ uten at $\mathcal{R}$ terminerer da det fins en uendelig omskrivningssekvens $f(a,a)\xrightarrow[\mathcal{R}]{}b\xrightarrow[\mathcal{R}]{}f(a,a)\xrightarrow[\mathcal{R}]{}b\xrightarrow[\mathcal{R}]{}\cdots$. Eksemplet motiverer også i viss grad det lange beviset av det tilsynelatende opplagte teoremet 6.15.

Er det egentlig noen fordel med Waldmann-kravet i stedet for (pre-) regularitet? Det kunne virke litt unødvendig å ha deklarasjonene $a:s_1$ og $a:s_2$. Vil man at tolkningen av a skal være med i $\mathcal{A}_{s_1}$ og $\mathcal{A}_{s_2}$ for alle Σ-algebraer $\mathcal{A}$ kunne man vel like gjerne innført en ny type $s_1 \diamond s_2 \leq s_1, s_2$ og latt $a : s_1 \diamond s_2$?

Uansett er Waldmanns innfallsvinkel litt spesiell, og det gir problemer med å sjekke typeminskning i "Waldmann-signaturer", samtidig som han som nevnt må forlange regularitet under komplettering. Derfor mener jeg at det ikke er noen stor svakhet ved TT-metoden at den ikke klarer av å behandle denne type systemer. Jeg tror også at man uten store problemer ville klare å lage en speiling også for Waldmann-systemer gitt beviset for lemma 2.5 i [Wal92].

Unøyaktige speilinger

TT1-, TT2- og TT3-metodene tar ikke hensyn til hvilke F-sprang som virkelig forekommer i et system, men hvilke som ut i fra signaturen *kunne* forekomme. Når f. eks. alle reglene er typebevarende oppstår aldri F-sprang slik at begrensningen $f_w \succsim_F f_{w'}$ er overflødig. Dette problemet, og det enkle forslaget til forbedring er tatt opp i avsnitt 8.8.3 og induserer TT4-, TT5-, ...-metoder. Man må som vanlig vurdere enkelhet og effektivitet opp mot styrken til metoden.

Treghet

Det er vanskelig å komme bort fra at TT3-metoden kan virke treg i store systemer, men å påvise terminering av komplekse systemer på en mer effektiv måte synes i høyeste grad å være en ikke-triviell oppgave.

12.3 Utfordringer

Hvilke forbedringer kan så gjøres på TT-fremgangsmåten?

Finne passende ordninger

Etter innføringen av forenklingsordningsbegrepet er det beskrevet fler utypede termineringsordninger enn det er mygg på Sognsvann en vakker sommerkveld. Det kunne derfor være en utfordring å finne de ordningene som passer best til å vise terminering av den spesielle type omskrivningssystemer som dannes av speilinger. I denne oppgaven har vi f. eks. sett at stiordningene passet dårlig på TT3-speilinger, slik at jeg måtte definere og bruke ordningen *ukbo* som passet bedre for disse.

Prøve å effektivisere termineringstesten

Det er sannsynligvis mange måter man kan prøve å effektivisere metoden på. Jeg tenkte blant annet på å utnytte at $\lfloor l\mu \rfloor \longrightarrow \lfloor r\mu \rfloor$ er på samme "form" som $\lfloor l \rfloor \longrightarrow \lfloor r \rfloor$ for alle spesialiseringer μ. Et enkelt eksempel viser at man kan utnytte egenskaper ved ordninger slik at $\lfloor l\mu \rfloor \succ \lfloor r\mu \rfloor$ følger automatisk når man har vist $\lfloor l \rfloor \succ \lfloor r \rfloor$.

Anta f. eks. at $we(f_w) = we(f_{w'})$ gjelder for alle Σ-funksjonssymboler f ved bruk av termineringsordningen *ukbo*. Har man da klart å vise $vekt(\lfloor l \rfloor) > vekt(\lfloor r \rfloor)$, gjelder også $vekt(\lfloor l\mu \rfloor) > vekt(\lfloor r\mu \rfloor)$ og dermed $\lfloor l\mu \rfloor \succ_{ukbo} \lfloor r\mu \rfloor$ for alle μ, slik at man slipper å bruke tid på å vise dette eksplisitt.

TT3-metoden forlanger forholdsvis mye testing, slik at det sannsynligvis er en del å hente på å effektivisere testene.

Forandre den typeordnede regelmengden

Kapittel 11 viste at TT-metoden kunne gjøres sterkere ved først å forandre den typeordnede regelmengden. Det kunne være en utfordring å forandre omskrivningssystemet slik at det lettest mulig å vise terminering, men dette hører hører kanskje like mye til programtransformasjonsteori?

Finne bedre speilinger

Det kan som vist i avsnitt 8.8.3 være mye å tjene på å lage mer nøyaktige speilinger. Ovennevnte avsnitt beskriver hvordan dette kan gjøres, og kanskje kan man få en perfekt speiling?

Kapittel 13

Nøkkelen til typeordnede termineringsordninger

Selv om de geniale løsningene på vitenskapelige problemer sjelden dukker opp der man tror de vil gjøre det, forsøker jeg i dette kapitlet forutsi hvor man skal lete for å finne bedre måter enn TT-metoden for å påvise typeordnet terminering. Jeg vil også si litt om hvilke idéer man ikke trenger å forfølge.

13.1 Håndtere variable

Jeg har under arbeidet med denne oppgaven og særlig i mine forsøk på å finne en perfekt speilingsalgoritme blitt overbevist om at den eneste *fundamentale* forskjellen mellom utypet og typeordnet omskrivning ligger i variablene.

I en utypet regel kan en variabel x instansieres med en hvilken som helst term i termuniverset. Enhver termineringsordning $\succ$ må ta hensyn til dette, og en variabel kan derfor aldri sammenlignes i $\succ$ med en term som ikke inneholder den samme variabelen. I stiordningene oppnås dette ved at en variabel ikke er sammenlignbar i $\succsim_F$ med noe annet symbol, og i *ukbo* sørger vi for at variabelen har samme vekt som minste mulige grunnterm, samtidig som vi forlanger $n(x,t) \geq n(x,t')$ av $t \succ_{ukbo} t'$.

En typet variabel $x{:}s$ i en typeordnet regel kan derimot for det meste *ikke* substitueres med alle termer i $\mathcal{T}(\Sigma, X)$. Derfor skulle det ikke være utenkelig at $x{:}s$ på én eller annen måte kunne sammenlignes med en term t der $t \notin \mathcal{T}(\Sigma, X)_s$. Følgende eksempel motiverer en presedens $x \succ_F a$ ved av bruk av $\succ_{mpo}$ direkte på et typeordnet system.

Eksempel 13.1 La den regulære signaturen Σ og den typeordnede regelmengden $\mathcal{R}$ være

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \longrightarrow s$
	$a : s$
	$b : s'$
Regel:	$(\forall x{:}s')\ f(x) \longrightarrow f(a)$

Variabelen x kan ikke instansieres med a, så $\mathcal{R}$ terminerer (uavhengig av eventuelle nye deklarasjoner i Σ). I *mpo* burde man derfor muligens tillate en presedens $x \succ_F a$ for å kunne vise $f(x) \succ_{mpo} f(a)$, og i *ukbo* kunne kanskje en vektlegging $we(x) > we(a)$ aksepteres?

Dette er vel og bra, men anta at vi legger til en (typeminskende) regel $a \longrightarrow b$ til $\mathcal{R}$, som nå blir ikke-terminerende da $f(b) \xrightarrow[\mathcal{R}]{} f(a) \xrightarrow[\mathcal{R}]{} f(b) \xrightarrow[\mathcal{R}]{} \cdots$ er en uendelig omskrivningssekvens. Med presedensen $x \succ_F a$ og $a \succ_F b$ ville man likevel kunne vise $\succ_{mpo}$ -minskning av begge reglene. Feilen her er at x kan substitueres med b, og ifølge diskusjonen over burde x og b være usammenlignbare i $\succ_{mpo}$, noe de ikke er da $x \succ_F a$ og $a \succ_F b$ impliserer $x \succ_F b$ og dermed $x \succ_{mpo} b$.

Merk forøvrig at TT3-*mpo* kan vise terminering av $\{f(x:s) \longrightarrow f(a)\}$ når F settes lik $\{a, b, f_s, f_{s'}\}$.

13.1.1 Regler for sammenligning av variable i termineringsordninger

Det er altså ønskelig å kunne sammenligne en typeordnet variabel med andre termer i termineringsordninger. Gitt en monoton, transitiv og ikke-refleksiv (utypet) termineringsordning $\succ$ ser jeg på noen betingelser som må være oppfylt for at typeordnet variabel $x:s$ skal være sammenlignbar med en annen typeordnet term t i $\mathcal{T}(\Sigma, X)$. Egenskapen man hele tiden må beholde er

$$t \succ t' \Longrightarrow t\sigma \succ t'\sigma$$

for alle typeordnede substitusjoner $\sigma : X \rightarrow \mathcal{T}(\Sigma, Y)$. Vi må derfor forlange

$$\forall t \in \mathcal{T}(\Sigma, X) \mid x \succ t \Longrightarrow x\sigma \succ t\sigma \tag{13.1}$$

og

$$\forall t \in \mathcal{T}(\Sigma, X) \mid t \succ x \Longrightarrow t\sigma \succ x\sigma \tag{13.2}$$

for alle slike typeordnede substitusjoner $\sigma : X \rightarrow \mathcal{T}(\Sigma, Y)$. I mange interessante tilfeller vil termen t i betingelsene 13.1 og 13.2 *ikke* inneholde variable og betingelsene kan da skrives hhv.

$$x:s \succ t \Longrightarrow (\forall t' \in \mathcal{T}(\Sigma, X)_s \mid t' \succ t)$$

og

$$t \succ x:s \Longrightarrow (\forall t' \in \mathcal{T}(\Sigma, X)_s \mid t \succ t')$$

Merk at kravene 13.1 og 13.2 sammen med ikke-refleksivitet av $\succ$ impliserer

$$\forall t \in \mathcal{T}(\Sigma, X)_s \mid x:s \notin \mathcal{V}(t) \Longrightarrow (x \not\succ t \wedge t \not\succ x)$$

ved substitusjonen $\sigma = \{x \mapsto t\}$.

13.1.2 Typeordnet *mpo*

For enkelhets skyld ser jeg i resten av dette avsnittet på den "gamle" ordningen $\succ_{mpo}$, dvs. den som er gitt i definisjon 2.18 på side 28. Vi kan nå bruke ordningen $\succ_{mpo}$ der det tidligere kravet om at en variabel ikke måtte være sammenlignbar i $\succ_F$ med noe annet symbol erstattes med de to betingelsene 13.1 og 13.2 over. Denne nye varianten av $\succ_{mpo}$ som jeg kaller *typeordnet mpo* og skriver $\succ_{tmpo}$, kan brukes til å vise terminering av typeordnede omskrivningssystemer.

Proposition 13.1 *For alle termer t, t' og typeordnede substitusjoner $\sigma : Y \rightarrow \mathcal{T}(\Sigma, X)$ gjelder*

$$t \succ_{tmpo} t' \Longrightarrow t\sigma \succ_{tmpo} t'\sigma$$

når $\succ_F$ er slik at $\succ_{tmpo}$ tilfredsstiller kravene 13.1 og 13.2.

Bevis: Beviset foregår rett frem ved induksjon på størrelsen av beviset for $t \succ_{tmpo} t'$ og siden beviset er uten vanskeligheter utelates det her. □

Jeg har bare forandret presedenskravet i $\succ_{tmpo}$, ikke selve ordningen $\succ_{mpo}$. Ordningen $\succ_{tmpo}$ vil derfor alltid være velfundert når $\succ_{mpo}$ er det, og kan derfor vise terminering av typeordnede systemer.

Eksempel (forts. av eks. 13.1) La $\succ_F$ være gitt ved $x : s' \succ_F a$ og $b \succ_F a$. Jeg skal vise at $\succ_{tmpo}$ oppfyller betingelsene 13.1 og 13.2 med denne presedensen.

Krav 13.1 x er bare større enn a i $\succ_{tmpo}$. Alle termer av typen s', dvs. termen b må da være større enn a, og det stemmer også.

Krav 13.2 De eneste termene som er større enn x er på formen $t[x]_p$ der $p \neq \epsilon$. Betingelse 13.2 er tilfredsstilt siden $t[x\sigma]_p \succ_{tmpo} x\sigma$.

Siden $f(x : s') \succ_{tmpo} f(a)$ har vi dermed påvist terminering av $\mathcal{R}$. Merk også at presedensen $a \succ_F b$ er utelukket, slik at det umulig å feilaktig "påvise" terminering av $\mathcal{R} \cup \{a \longrightarrow b\}$.

Eksemplet brukte en presedens $x \succ_F a$, men terminering kunne også vises med TT3-*mpo*. Hverken TT3-*mpo* eller TT3-*ukbo* kan derimot vise terminering av systemet i følgende eksempel som også illustrerer bruk av den motsatte presedensen $a \succ_F x$.

Eksempel 13.2 La omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være gitt

Typer:	s, s'
	$s' \leq s$
Funksjoner:	$f : s \rightarrow s$
	$g : s, s \rightarrow s$
	$a : s$
	$b : s'$
Regel:	$(\forall x : s')\ g(f(a), x) \longrightarrow g(f(x), x)$

$\mathcal{R}$ terminerer da x bare kan instansieres med b. Den mest nøyaktige syntaktiske speilingen av regelen er $g_{s,s'}(f_s(a), x) \longrightarrow g_{s,s'}(f_{s'}(x), x)$, som hverken er $\succ_{mpo}$- eller $\succ_{ukbo}$-minskende (i de fleste utypede ordninger koker det hele ned til å vise det umulige $f_s(a) \succ f_{s'}(x)$). TT-*mpo*- og TT-*ukbo*-metodene må derfor gi opp her.

La så $\succ_{tmpo}$-presedensen $\succ_F$ være gitt $a \succ_F x : s'$ og $a \succ_F b$. Det første kravet til $\succ_{tmpo}$ er trivielt oppfylt da ingen term er $\succ_{tmpo}$-mindre enn x. Siden a er den minste termen som er større enn x, og $a \succ_{tmpo} b$ gjelder så er også betingelse 13.2 tilfredsstilt. Med den gitte presedensen kan man vise $f(g(a), x) \succ_{tmpo} f(g(x), x)$ og dermed terminering av $\mathcal{R}$.

13.1.3 Her er altså nøkkelen – og utfordringene

Nøkkelen til hele termineringsproblemet i typeordnet omskrivning ligger med stor sannsynlighet i å håndtere variablene på en fornuftig måte. Jeg skulle tro at man må inn med en helt ny tankegang om termineringsordninger, men dette er utgangspunktet. Jeg illustrerte nytten av variabeltypeanalyse veldig grovt med å modifisere presedenskravene til stiordningene og kalte ordningen $\succ_{mpo}$ med disse nye kravene for $\succ_{tmpo}$. Selv om ordningen $\succ_{tmpo}$ bare var ment som et eksempel og følgelig var veldig enkel, og neppe vil bli brukt i praksis, viste det seg at den i noen systemer kunne vise terminering der TT3-*ukbo* og TT3-*mpo* kom til kort.

Kanskje TT-metoden kan kombineres med én eller annen form for analyse av hva en variabel kan instansieres med?

13.2 Her er det ingen nøkler

13.2.1 Ordninger over typer

"Lag algoritmer som påviser typeordnet terminering!". Gitt denne oppgaven består sannsynligvis den aller første idéen man får i å bruke en leksikografisk kombinasjon $(\succsim_{type}, \succsim_{term})$ der $\succsim_{type}$ sammenligner to termer med hensyn på "typestrukturen" i termene, mens $\succsim_{term}$ er en vanlig (utypet) termineringsordning over termer. Denne idéen prøves illustrert i følgende eksempel:

Eksempel 13.3 La omskrivningssystemet $\langle \Sigma, \mathcal{R} \rangle$ være gitt ved

Typer:	s, s_1, s_2
	$s_1 \leq s$
	$s_2 \leq s$
Funksjoner:	$g : s, s \longrightarrow s_2$
	$f : s \longrightarrow s$
	$f : s_1 \longrightarrow s_1$
	$a : s_1$
	$b : s_1$
Regler:	$(\forall x : s_1)\ \ f(x) \longrightarrow f(g(x,a))$
	$a \longrightarrow b$

Gitt en presedens $\succsim_S$ over typene i Σ (denne må ikke forveksles med relasjonen $\leq$) kunne ordningen $\succsim_{type}$ defineres

type-1: $$\frac{LS(t) \succ_S LS(t')}{t \succsim_{type} t'}$$

type-2: $$\frac{LS(f(t_1,\ldots,t_m)) = LS(g(t'_1,\ldots,t'_n)) \qquad \langle t_1,\ldots,t_m\rangle \succsim^{lex}_{type} \langle t'_1,\ldots,t'_n\rangle}{f(t_1,\ldots,t_m) \succsim_{type} g(t'_1,\ldots,t'_n)}$$

På denne måten er begge reglene $(\succsim_{type}, \succsim_{mpo})$-minskende siden $f(x:s_1) \succ_{type} f(g(x,a))$ med typepresedens $s_1 \succ_S s_2$, mens $a \approx_{type} b$ samtidig som $a \succ_{mpo} b$ med presedens

$a \succ_F b$. Det bør bemerkes at ordningen $\succsim_{type}$ definert her bare er ment som et eksempel og absolutt ikke er velfundert.

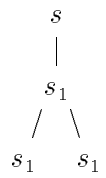
Kunne man finne en velfundert ordning $\succsim_{type}$ over "trær av typer", kunne en slik leksikalsk kombinasjon $(\succsim_{type}, \succsim_{term})$ brukes. Mitt råd er imidlertid:

Glem hele idéen!

Begrunnelsen er at jeg føler meg meget sikker på at dersom det fins en slik kombinasjon $(\succsim_{type}, \succsim_{term})$ så vil det alltid finnes en leksikografisk kombinasjon av to ordninger $(\succsim', \succsim_{term})$, der også $\succsim'$ er en ordning over termer, som viser termineringer av speilinger.

Anta nemlig at det fins en presedens $\succsim_S$ over typer og en velfundert ordning $\succsim_{type}$ over trær av typer. La oss så bruke en speiling der jeg merker alle funksjonssymbolene med minste type istedenfor minste aritet. Jeg har i kapittel 6 begrunnet at dette gir en i termineringshenseende dårligere speiling enn om alle symboler merkes med tilhørende aritet. I eksempelet over ville termen $f(g(x,a))$ altså blitt merket

mens den samme termen representert som et tre av typer ville blitt



Så definerer jeg en presedens $\succsim_F$ over disse merkede funksjonssymbolene ved $f_s \approx_F g_s$ for alle Σ-symboler f, g og $s_1 \succsim_S s_2 \Longleftrightarrow f_{s_1} \succsim_F g_{s_2}$ for alle typer s_1 og s_2 og funksjonssymboler f, g i F. Termordningen $\succsim'$ defineres da på samme måte som $\succsim_{type}$ der relasjonen $\succsim_S$ erstattes med $\succsim_F$. Den observante leser har sett at dette ikke nødvendigvis går bra, siden vi i den merkede termen har en variabel x, når vi samtidig har en konstant s_1 i "typerepresentasjonen". Dette er for så vidt riktig, men s_1 står også for en variabel, slik at man også i $\succsim_{type}$ må være meget forsiktig med denne. Hvilken ordning $\succsim_{type}$ over "typetrær" ville f. eks. kunne fastslå $f(x:s_1) \succsim_{type} f(g(x,a))$ i forrige eksempel, samtidig som den *ikke* ville hatt $f(x:s_1) \succsim_{type} f(g(f(x),a))$ siden denne siste regelen forårsaker ikke-terminering?

Merk også at at man heller ikke slipper F-sprang (som her må bli kalt S-sprang!) når man bare ser på termene som trær av typer.

Jeg er derfor nesten helt sikker på at $\succsim'$ definert på denne måten ville være like bra som $\succsim_{type}$. Er den ikke dét får man heller gå løs på utfordringene med å håndtere variable i termineringsordningene. Konklusjonen må derfor bli at det for enhver leksikografisk sammenligning $(\succsim_{type}, \succsim_{term})$ alltid kan kan finnes en like bra termordning $(\succsim', \succsim_{term})$ over termer i speilinger.

13.2.2 Andre idéer som ikke må forfølges

Vi har tidligere sett at minimalisering aldri gir bedre resultater enn TT-metoden. Jeg ser for øyeblikket ikke for meg så mange lure idéer, men de som har gode idéer kan også først lese avsnitt 6.7 på side 95 hvor jeg begrunner fremgangsmåten ved speilinger og tilbakeviser noen smarte forsøk på å utnytte typer i termineringsordninger.

Kapittel 14

Oppsummering

Denne oppgaven hadde to målsetninger. Den sekundære var å gi en kort forklarende innføring i typeordnet omskrivning, noe jeg gjør i den første forberedende delen av oppgaven. Her ser jeg også på hvordan retrakter kan brukes i omskrivningssystemer og introduserer begrepet *slappe deklarasjoner* som jeg vil ha behov for i hoveddelen, del II, av oppgaven.

Hovedhensikten var å utforske det forholdsvis ukjente terrenget "terminering av typeordnet omskrivning", en hvit flekk på kartet over den ellers godt beskrevne omskrivningsverdenen. Jeg tok utgangspunkt i Gnaedig [Gn92a], og prøvde først å generalisere idéen med minimale signaturer. Dette viste seg å være like lurt som å dra vestover for å finne sjøveien til India.

I stedet beholdt jeg Gnaedigs idé med å merke funksjonssymboler og brukte, etter å ha definert typeordnet terminering, disse merkede funksjonssymbolene til å lage en speiling, en utypet projeksjon av et typeordnet omskrivningssystem. Man kan så benytte ordningene beskrevet i kapittel 2 for å vise at speilingen, og dermed det opprinnelige typeordnede systemet, terminerer. Jeg eksemplifiserte dette med å beskrive metoden TT1-*mpo*. Egenskaper ved mange termineringsordninger gjorde at jeg måtte ta i bruk kvasiordninger for å forbedre metoden. TT2-metoden jeg da fikk er ekte sterkere enn både Gnaedigs metode og en utypet analyse, men kunne forbedres videre til en mer komplisert og beregningskrevende variant, TT3. Stiordningene klarte ikke å illustrere det fulle potensialet til denne metoden. Imidlertid kan man bruke TT3-metoden med andre ordninger, f. eks. polynomiske ordninger. I denne oppgaven har jeg illustrert dette poenget ved å bruke metoden på en utvidelse av Knuth-Bendix-ordningen som jeg definerte i kapittel 9.

TT-metoden forlanger (pre-)regulære signaturer og typeminskende omskrivningsregler, men jeg skisserte i kapittel 12 hvordan metoden kan utvides til å kunne brukes også på ikke-typeminskende systemer. I kapittel 12 vurderer jeg også TT-metoden og forsøker å peke på videre utfordringer knyttet til metoden.

I kapittel 11 viste jeg at TT-metoden i noen spesielle tilfeller kan gjøres sterkere ved at man først transformerer regler i det typeordnede systemet og deretter bruker TT-metoden på det transformerte systemet.

Jeg avsluttet oppgaven med å løsrive meg fra TT-metoden og argumenterte for at ordninger over typestrukturen i termene sannsynligvis ikke er veien å gå i jakten på bedre metoder. I tillegg spådde jeg at nøkkelen til å finne andre måter å påvise typeordnet terminering ligger i å ta hensyn til at en variabel ikke kan instansieres med alle termene i et typeordnet

termunivers. Dette ble eksemplifisert med en ordning $\succ_{tmpo}$ som tillater at en variabel er sammenlignbar i ordningen med termer som variabelen ikke kan instansieres med. Med denne enkle forandringen av presedenskravene til ordningen $\succ_{mpo}$ kunne jeg vise terminering av et system der ingen speilingsmetode kan benyttes. Vil man bruke andre fremgangsmåter enn speilinger for å vise typeordnet terminering tror jeg man bør lete her, men på mitt kart over dette spennende området står det fremdeles "Here be Dragons".

Referanser

[CH93] H. Chen og J. Hsiang. *Order-Sorted Equational Specification and Completion.* Upublisert notat, Department of Computer Science, State University of New York at Stony Brook, 1993.

[Dahl92] O.-J. Dahl. *Verifiable Programming.* Prentice Hall, 1992.

[Der82] N. Dershowitz. *Orderings for term-rewriting systems.* Theoretical Computer Science 17(3):279-301, 1982.

[Der87] N. Dershowitz. *Termination of rewriting.* Journal of Symbolic Computation, 3:69-116, 1987.

[DJ90] N. Dershowitz og J.P. Jouannaud. *Rewrite systems.* Handbook of Theoretical Computer Science, kapittel 15, North-Holland, 1990.

[FGJM85] K. Futatsugi, J. A. Goguen, J-P. Jouannaud og J. Meseguer. *Principles of OBJ2.* Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages, Association for Computing Machinery 1985.

[GKK87] I. Gnaedig, C. Kirchner og H. Kirchner. *Equational Completion in Order-Sorted Algebras. Extended Abstract.* Ph.D.thesis, 1987.

[GKK90] I. Gnaedig, C. Kirchner og H. Kirchner. *Equational Completion in Order-Sorted Algebras.* Theoretical Computer Science 72:169-202, 1990.

[GM88] J. A. Goguen og J. Meseguer. *Order-Sorted Algebra I.* Center for the Study of Language and Information, Stanford University, 1988.

[Gn92a] I. Gnaedig. *Termination of order-sorted rewriting.* Proc. Algebraic and Logic Programming. Third International Conference, Springer-Verlag, 1992.

[Gn92b] I. Gnaedig. *ELIOS-OBJ: theorem proving in a specification language.* Proc. ESOP '92, 4th European Symposion on Programming, Springer-Verlag, 1992.

[GW88] J. A. Goguen og T. Winkler. *Introducing OBJ3.* Teknisk rapport, SRI International, Computer Science Lab., 1988.

[HKK94] C. Hintermeier, C. Kirchner og H. Kirchner. *Dynamically-typed computations for order-sorted equational presentations.* Utkast. Kommer i Proceeding of ICALP'94.

[JLR83] J-P. Jouannaud, P. Lescanne og F. Reining. *Recursive Decomposition Ordering.* I D. Bjørner (ed), Formal Descriptions of Programming Concepts – II, North-Holland 1983.

[KB70] D. Knuth og P. Bendix. *Simple word problems in universal algebras.* J. Leech, editor, Computational Problems in Abstract Algebra, side 263-297, Pergamon Press, Oxford, 1970.

[Kir92] B. Kirkerud. *Forelesningsnotater til kurset in307.* Institutt for informatikk, Universitetet i Oslo, 1992.

[Lin92] H. Linnestad. *Fatale skritt under komplettering.* Hovedoppgave, Institutt for informatikk, Universitetet i Oslo, 1992.

[Pla93] D. A. Plaisted. *Equational reasoning and term rewriting systems.* D. M. Gabbay, C. J. Hogger og J. A. Robinson, editors, Logical Foundations, kapittel 5. Oxford University Press, 1993.

[Rus87] M. Rusinowitch. *Path of Subterms Ordering and Recursive Decomposition Ordering Revisited.* Journal of Symbolic Computation, 3, 1987.

[Wal92] U. Waldmann. *Semantics of order-sorted specifications.* Theoretical Computer Science 94(1992) 1-35.